Universidade de Aveiro
2011
Departamento de Electrónica, Telecomunicações e Informática

João Filipe Lopes Seabra Lourenço

Implementação em FPGA da Modulação/Desmodulação OFDM 802.11p

Universidade de Aveiro
2011
Departamento de Electrónica, Telecomunicações e Informática

João Filipe Lopes Seabra Lourenço

Implementação em FPGA da Modulação/Desmodulação OFDM 802.11p

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira e Doutor João Nuno Pimentel da Silva Matos, Professores do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro.

O júri

Presidente	Prof. Doutor José Alberto Gouveia Fonseca Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Arguente	Prof. Doutor António José Duarte Araújo Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto
Orientadores	Prof. Doutor Arnaldo Silva Rodrigues de Oliveira Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
	Prof. Doutor João Nuno Pimentel da Silva Matos Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Agradecimentos

Agradeço à minha família e amigos por sempre me terem apoiado ao longo da minha vida académica.

Um agradecimento aos meus orientadores, por me terem dado a oportunidade de realizar esta dissertação, em especial ao Professor Doutor Arnaldo da Silva Rodrigues Oliveira que acompanhou o meu trabalho ao longo do projecto.

Um agradecimento especial ao Mestre Nelson Cardoso, pela ajuda prestada ao longo do trabalho.

Agradeço à Universidade de Aveiro, e em especial ao departamento de Electrónica, Telecomunicações e Informática, pelas excelentes condições de estudo que me proporcionaram.

A todos muito Obrigado!

Palavras-chave VHDL, OFDM, FPGA, IEEE 802.11p, DSRC, Comunicações Veiculares.

Resumo

As comunicações veiculares começam a despertar cada vez mais interesse, muito devido às novas potencialidades oferecidas pelo desenvolvimento das comunicações sem fio. Uma das mais promissoras tecnologias dedicadas à área das comunicações veiculares é o DSRC 5.9 GHz, que associado com a recente norma IEEE 802.11p é capaz de proporcionar comunicações fiáveis e robustas entre veículos.

Neste documento são mostradas as potencialidades desta tecnologia, e é apresentada a implementação de parte do sistema multiportadora OFDM, que constitui a camada física da norma 802.11p. Esta implementação faz-se recorrendo a circuitos reconfiguráveis de alta capacidade conhecidos como FPGA, e ao conceito de SDR (Software Defined Radio).

.

Keywords VHDL, OFDM, FPGA, IEEE 802.11p, DSRC, Vehicular Comunications.

Abstract

The vehicular communications beginning to attract an increasing interest, much of this caused by the new possibilities offered by the development of wireless communications. One of the most promising technologies in the area of vehicular communications is the DSRC 5.9 GHz, which associated with the recent standard IEEE 802.11p is able to provide reliable and robust communications between vehicles.
In this document are shown the potential of this technology, and is shown a partial implementation of the multicarrier system OFDM, which form the physical layer of standard 802.11p. This implementation is realized with the using of programmable circuits of high capability, known by FPGAs, and utilize the concept of SDR (Software Defined Radio).

Índice

Lista de figuras

Lista de tabelas

Acrónimos

A/D - Analógico/Digital

ASIC - *Application-Specific Integrated Circuit*

BER - *Bit Error Rate*

BPSK - *Binary Phase Shift Keying*

CR - Cognitive Radio

D/A - Digital Analógico

DSP - *Digital Signal Processor*

DSRC - *Dedicated Short Range Communications*

FDM - *Frequency Division Multiplexing*

FFT - *Fast Fourier Transform*

FIFO - *First In First Out*

FPGA - *Field Programmable Gate Array*

HDL - *Hardware Description Language*

ICI - *Inter Carrier Interference*

IEEE - *Institute of Electrical and Electronics Engineers*

IFFT- *Inverse Fast Fourier Transform*

IPcore - Intellectual Property core

ISE - *Integrated System Environment*

ISI - *Inter Symbol Interference*

MAC - *Medium Access Control*

MANET- *Mobile Ad-hoc NETwork*

OBU - *On Board Unit*

OFDM - *Orthogonal Frequency Division Multiplexing*

OSI - Open System Interconnection model

PAPR - *Peak to Average Power Ratio*

PHY - *Physical layer*

QAM - *Quadrature Amplitude Modulation*

QPSK - *Quadrature Phase Shift Keying*

RFID - Radio Frequency Identification

RSU - *Road Side Unit*

SDR - *Software Defined Radio*

V2I - *Vehicle to Infrastructure*

V2V - *Vehicle to Vehicle*

VANET- *Vehicular Ad-hoc NETwork*

VHDL - (*Very high speed integrated circuits) Hardware Description Language*

VSA - *Vector Spectrum Analyzer*

WAVE - *Wireless Access in Vehicular Environment*

1 Introdução

1.1 Enquadramento

O trânsito rodoviário tem vindo a aumentar grandemente nas últimas décadas, devido ao crescimento da população, ao desenvolvimento da economia, e muito devido ao estilo de vida actual, que obriga ao deslocamento de pessoas e mercadorias diariamente, e para isto, o uso do automóvel é o método por excelência para estas deslocações. O aumento do trânsito causa problemas facilmente visíveis todos os dias, por exemplo, nos grandes centros urbanos, como sejam os congestionamentos das vias que fazem perder tempo e paciência dos automobilistas. Como tal, é de todo o interesse utilizar todos os meios possíveis, para tornar as viagens de automóvel mais seguras e aprazíveis. Nesta área as comunicações sem fios podem dar uma grande ajuda. Com evolução que as comunicações sem fio tem sofrido nas últimas décadas, é hoje em dia possível falar-se no conceito de comunicações veiculares. As comunicações veiculares permitem que os elementos da via (automóveis, autocarros, pesados de mercadorias, a própria via e seus constituintes), possam operar em rede e trocar informação entre si.

Estas informações podem passar por, avisos de acidente ou incidentes na via, informações sobre o congestionamento do trânsito, alertas sobre as condições meteorológicas e estado da via, e sendo estas informações passadas em tempo real, criar um paradigma completamente novo no que respeita a segurança rodoviária. Além destas potencialidades, outros cenários impensáveis há poucos anos, são agora um objectivo a atingir pelas comunicações entre veículos (V2V, *Vehicle to Vehicle*), e entre veículo e infra-estrutura (*V2I, Vehicle to Infrastructure*), como sejam informação baseada em localização, *social networking*, e até jogos interactivos. Esta tecnologia pode vir a ser implementada em novos dispositivos, ou pode também ser usada como complemento a dispositivos já existentes, como sejam o GPS que recorrendo às informações em tempo real sobre possíveis condicionantes na via, poderão fazer uma escolha do trajecto mais adequado.

Para suportar estas aplicações, novos desafios à tecnologia são apresentados, pois para além de todos os problemas que uma habitual rede móvel sem fio, também conhecida por MANET (*Mobile Ad-hoc Network*) tem de ultrapassar, somam-se a estes as condições específicas a um ambiente automóvel, onde os nós que formam as novas redes, conhecidas por VANET (*Vehicular Ad-hoc NETwork*), ou seja os próprios automóveis, se deslocam a velocidades muito elevadas, e ainda as condições adversas do canal de comunicação num ambiente rodoviário. Desta forma os standards de comunicações usados nas em MANETs, não são suficientemente robustos para comportar estas novas redes (VANET). Assim nos últimos anos tem vindo a ser desenvolvidas diversas normas e protocolos dedicados a este tipo de comunicações. Uma destas normas é a IEEE

802.11p, na qual são definidas as camadas física (PHY), e controlo de acesso ao meio (MAC), para comunicações veiculares DSRC (*Dedicated Short Range Communications*) na banda dos 5.9GHz. O DSRC é um termo que era normalmente usado para descrever comunicações de curto alcance entre veículos e a infra-estrutura (e.g pagamento electrónico de portagens), mas em 1999 quando nos Estados Unidos o *US Federal Communication Commission* alocou 75MHz do espectro na banda dos 5.9GHz, estas passaram a ser conhecidas como DSRC 5.9GHz WAVE (*Wireless Acess in Vehicular Environment*), e hoje em dia é uma das tecnologias que tem sido alvo de desenvolvimento com vista à implementação de VANETs, tendo a Europa adoptado também esta banda.

Por outro lado o surgimento das comunicações veiculares só é possível devido à enorme evolução na tecnologia microeletrónica nas últimas décadas. È hoje em dia possível ter circuitos integrados com centenas de milhões de transístores. Este nível de desenvolvimento a nível dos circuitos permitiu chegar a dispositivos lógicos de elevada complexidade e reconfiguráveis, conhecidos como FPGAs. Estes dispositivos permitem hoje em dia a implementação de sistemas completos num único chip, podendo agregar numa só FPGA, desde processadores dedicados, co-processadores, periféricos específicos para o sistema, memórias, entre outros. As FPGAs disponíveis hoje em dia têm capacidades equivalentes a milhões de portas lógicas, que se dividem tanto em recursos programáveis, como em recursos fixos tais como LUTs, células DSP, processadores de uso geral (e.g, PowerPC, MicroBlaze), controladores para protocolos de comunicação, como por exemplo Ethernet. Isto associado a linguagens de descrição de hardware (HDL), tais como o VHDL ou o VERILOG, tornam a fase de prototipagem de um sistema complexo, muito mais rápida e flexível.

Em grande parte associado ao grande desenvolvimento das FPGAs, nos últimos anos tem-se desenvolvido um novo conceito para os rádios, conhecido como: rádios definidos por software (SDR- *Software Defined Radio*). Este conceito consiste, muito resumidamente, em que os rádios, idealmente sejam constituídos por uma antena seguida de conversores analógico-digital/digital-analógico (AD/DA), e que todo o processamento dos sinais rádio sejam feitos em software, em processadores de uso geral ou FPGAs ou DSPs, ou até mesmo e idealmente em computadores de uso pessoal, e não em circuitos específicos para a tecnologia que se quer usar.

Este trabalho vai portanto abordar os pontos indicados, ou seja as comunicações veiculares, em particular o DSRC 5.9GHz WAVE, as FPGAs e o *software defined radio*.

1.2 Motivação

O crescimento do interesse no ramo automóvel na área das comunicações veiculares tem vindo a aumentar grandemente, devido às suas potencialidades, principalmente no que diz respeito a novos sistemas de segurança para os veículos. Assim associado o interesse económico ao avanço da tecnologia, e o interesse académico numa área ainda muito recente e com poucos resultados práticos alcançados, tem feito surgir parcerias entre as academias e as empresas do ramo, para desenvolver sistemas que possam ser integrados nos veículos e na infra-estrutura, de baixo custo e fiáveis. Isto recorrendo à utilização dos novos standards que têm surgido nos últimos tempos. Um desses projectos onde este trabalho está inserido, é o projecto HEADWAY [1], que é uma parceria entre a BRISA, o Instituto de Telecomunicações de Aveiro, e o Instituto Superior de Engenharia de Lisboa (ISEL). Este projecto tem como área de investigação o DSRC 5.9GHz WAVE, e as normas que o compõem, estando a minha parte de estudo focada numa parte da camada física da norma 802.11p. Mais especificamente a parte de mapeamento das portadoras, a inserção das portadoras piloto a IFFT, e a inserção do prefixo cíclico, isto na parte de modulação do sistema OFDM que constitui esta camada física. Na desmodulação do sistema OFDM, a parte de remoção do prefixo cíclico, a FFT e o desmapeamento das portadoras, são os elementos deste projecto que serão alvo do meu estudo. Associando esta tecnologia às vantagens que as FPGAs oferecem, (reconfigurabilidade, capacidade de integração), prevê-se obter um dispositivo de baixo custo, e reconfigurável, o que nesta fase em que as normas surgem e são alteradas constantemente, é uma grande vantagem. Nesta conjuntura este é um aliciante desafio no qual aceitei participar e dar o meu contributo.

1.3 Objectivos

Este trabalho tem como objectivos:

- Estudo das comunicações DSRC 5.9GHz WAVE, ao nível das aplicações, conceito e standards usados.
- Estudo de SDR (*Software Defined Radio*), ao nível do conceito, das aplicações, vantagens, limitações e desafios da tecnologia.
- Conhecimento dos dispositivos programáveis de alta densidade, mais propriamente as FPGAs da XILINX, bem como o software disponibilizado pela mesma marca para o desenvolvimento de projectos nos seus produtos (e.g ISE (Integrated Software Envieronment)).

- Estudo dos sinais OFDM, ao nível de aplicações, vantagens e funcionamento.

- Implementação em VHDL, e simulação faseada de alguns dos blocos que formam a camada física da norma IEEE 802.11p, dividida em transmissão e recepção.

- Integração e encapsulamento dos vários blocos, e simulação completa de toda a cadeia.

- Síntese e implementação em FPGA da camada física, e validação no hardware de cada um dos blocos.

- Configuração de testes na FPGA para a avaliação do funcionamento da camada física da norma IEEE 802.11p.

1.4 Organização da dissertação

Capitulo 1

No primeiro capítulo desta dissertação, pretende-se dar uma pequena introdução e enquadramento geral em que este trabalho foi desenvolvido, a motivação para o mesmo, e os objectivos a alcançar ao longo do projecto.

Capítulo 2

Neste capítulo são aprofundadas as comunicações veiculares, onde são apresentadas as tecnologias actuais usadas, os seus prós e contras, e onde se mostra aquilo que tem vindo a ser feito nesta área, i.e projectos desenvolvidos, em desenvolvimento etc. É também alvo de estudo neste capítulo a temática dos rádios definidos por software, o que são, quais as suas vantagens e aplicações, desvantagens, e onde se faz ligação com este projecto.

Capítulo 3

Aqui é apresentada a camada física da norma IEEE 802.11p, onde se mostra os seus blocos constituintes e quais as funcionalidades que esta irá implementar. Pretende-se ainda neste capítulo fazer uma introdução ao OFDM, visto ser parte constituinte do 802.11p.

Capítulo 4

O quarto capítulo está dedicado à implementação, testes, síntese e validação dos blocos individuais da camada física, na transmissão em OFDM.

Capítulo 5

O quinto capítulo está dedicado à implementação, testes, síntese e validação dos blocos individuais da camada física, na recepção em OFDM.

Capítulo 6

Este capítulo é mostrado o encapsulamento de toda a cadeia, sua síntese e implementação na FPGA, bem como os resultados obtidos com esta implementação.

Capítulo 7

O sexto e último capítulo fica dedicado às conclusões, e propostas de trabalho futuro.

2 Conceitos Fundamentais

2.1 Comunicações Veiculares

Espera-se que no futuro os elementos que constituem uma rodovia, automóveis, autocarros, pesados, a própria via de trânsito, comuniquem entre si. A estas comunicações dá-se o nome de comunicações veiculares. Estas comunicações dividem-se portanto entre comunicações entre veículos (V2V – *Vehicle to Vehicle*), e veículos e infra-estrutura (V2I – *Vehicle to Infrastructure*), estes são os tipos de comunicações que constituem as redes conhecidas por VANETs (*Vehicular Ad hoc NETworks*). VANETs são portanto redes ad hoc, cujos seus nós constituintes são os elementos da via, e que inerentemente não são fixos. Ou visto de uma maneira simplista, VANETs tem como conceito, pegar nas habituais e baratas WLANs (*Wireless Local Area network*), que conectam computadores entre si e à internet, e colocá-las nos automóveis para os ligar do mesmo modo. As principais aplicações destas redes visam: a segurança rodoviária, informações sobre condições de tráfego e até entretenimento, sendo actualmente as duas principais áreas de interesse, a segurança rodoviária, e a eficiência de tráfego. Para conseguir estas aplicações estas redes para além de todos os desafios que se apresentam nas habituais redes móveis sem fios, ou também chamadas MANETs (*Mobile Ad hoc NETwork*), somam-se a estes, as condições específicas de um ambiente rodoviário em que se tem de ter em conta as altas velocidades dos veículos [2] [3].

fig. 1 – Exemplo de comunicações veiculares (retirado de [4])

Estas condições adversas trazem grandes desafios à tecnologia, e como esta é uma área que desperta bastante interesse, pelo seu potencial económico, estando a despertar o interesse de grandes fabricantes de automóveis como por exemplo a norte americana *Daimler Chrysler,* ou as europeias *Mercedes-benz*, *BMW* ou *Volvo*, assim como de concessionárias de auto-estradas como a nacional Brisa, que é aliás um dos parceiros que está a desenvolver o projecto Headway [1] onde este trabalho se insere. Os esforços tem sido feitos para tornar as potencialidades das comunicações veiculares uma realidade. Uma das tecnologias que tem vindo a ser mais estudada e aplicada, e que mostra ser a mais promissora, é o DSRC 5.9GHz WAVE. Esta tecnologia utiliza a norma IEEE 802.11p na sua camada física, e vai ser essencialmente o objecto de estudo deste trabalho.

2.2 DSRC, 802.11p WAVE

DSRC é a sigla para *dedicated short range comunications,* em português comunicações dedicadas de curto alcance, é um protocolo desenvolvido inicialmente na década de 90 para comunicações entre veículos e infra-estrutura (V2I) sendo a sua utilização mais conhecida e utilizada, a cobrança electrónica de portagens ou estacionamentos. Esta tecnologia utilizava a banda dos 915 MHz e utilizava identificação por rádio frequência (RFID), mas em estudos que se seguiram à criação deste tipo de comunicações rapidamente levou a outras possibilidades, nomeadamente a troca de mensagens de segurança, que possam contribuir para a segurança rodoviária. Neste âmbito foi então criado um consórcio com o objectivo de desenvolver as DSRC em ambientes veiculares. Posteriormente com o aparecimento em 1997 da norma IEEE 802.11, chegou-se à conclusão que com a utilização deste na banda dos 5.9GHz, se poderia tirar partido do modo *ad-hoc* do 802.11, o que permitiria a comunicação entre veículos (V2V). Surge então o DSRC 5.9GHz para comunicações veiculares, também conhecido como WAVE. Com o crescente aumento do interesse das comunicações veiculares o IEEE, reuniu esforços e fecha em 2010, o standard IEEE 802.11p, que é baseado no IEEE 802.11a, mas este é dedicado especificamente às comunicações veiculares. O IEEE 802.11p define a camada física e MAC e juntamente com os standards IEEE 1609.x para as camadas superiores, formam então o conjunto de standards que o DSRC actual utiliza [5].

Recorrendo a dados referentes aos Estados Unidos da América, dados que se podem extrapolar para o resto do planeta, diz que em 1997 os acidentes rodoviários eram a quarta causa de morte nos EUA, ficando à frente de doenças como, diabetes, pneumonia e o vírus HIV [5]. Em outro estudo, temos a informação que só no ano de 2003, os acidentes rodoviários tiveram um custo monetário avaliado em 230 mil milhões de dólares, ao qual se junta o número de vítimas mortais de 42643! Se pensarmos que muitos destes acidentes poderiam ser evitados com a

simples troca de mensagens de alerta entre os elementos da via, vemos a extrema importância que as comunicações veiculares podem ter, e em especial as comunicações DSRC WAVE na segurança rodoviária.

Posto isto em 1999 o sistema de transportes inteligentes dos EUA (ITS), licenciou 75Mhz na banda dos 5.9GHz (5.850 – 5.925 GHz), para as DSRC.

2.2.1 Arquitectura e Características

As comunicações em DSRC, estão divididas em termos de Hardware em duas unidades, as unidades que se encontram nos veículos (OBU – *On Board Unit*) e pelas unidades fixas na via (RSU – *Road Side Unit*), estando as RSU ligadas tipicamente à rede do concessionário da via, enquanto que as OBU estão ligadas a rede interna do veiculo. Para a comunicação entre as duas unidades, o DSRC WAVE é utilizado.

Nos Estados Unidos, a largura de banda de 75 MHz reservada para o DSRC é actualmente formado por 7 canais de comunicação de 10MHz cada, que se dividem em canais de controlo, e canais de serviço como está ilustrado na figura seguinte

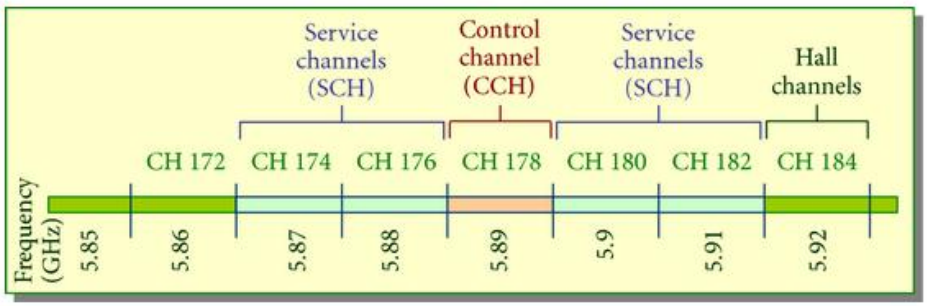

fig. 2 – Canais que compõem o DSRC (retirado de [5])

Estando o canal de HALL (*High Availability Low Latency*), guardado para utilizações futuras, o canal de controlo (CCH) é usado para comunicações seguras e para controlo, e os canais de de serviço são utilizados para serviços IP, por exemplo.

O *stack* com o modelo OSI completo de comunicações do DSRC pode ser visto na figura 3, onde se pode ver todos os standards que as compõem, desde a camada física até às camadas de aplicação. Podemos ver que camada física é constituída pelo IEEE 802.11.p, as seguintes camadas superiores reúnem vários standards baseados no IEEE 1609.x, podemos ver também que têm suporte para IPv6, para correr as aplicações.

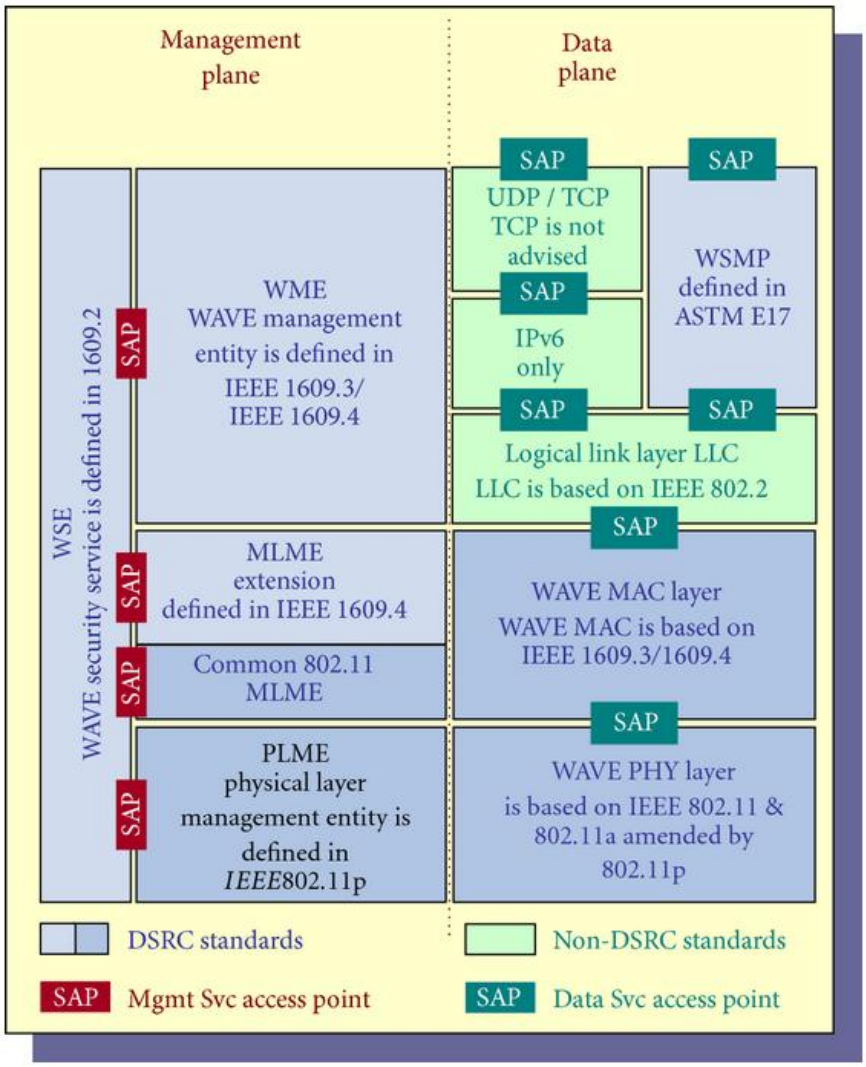

fig. 3 – Stack completo de comunicações DSRC (retirado de [5])

Uma vista geral de como o espectro dedicado ao DSRC se divide em todo o mundo pode ser visto na figura 4

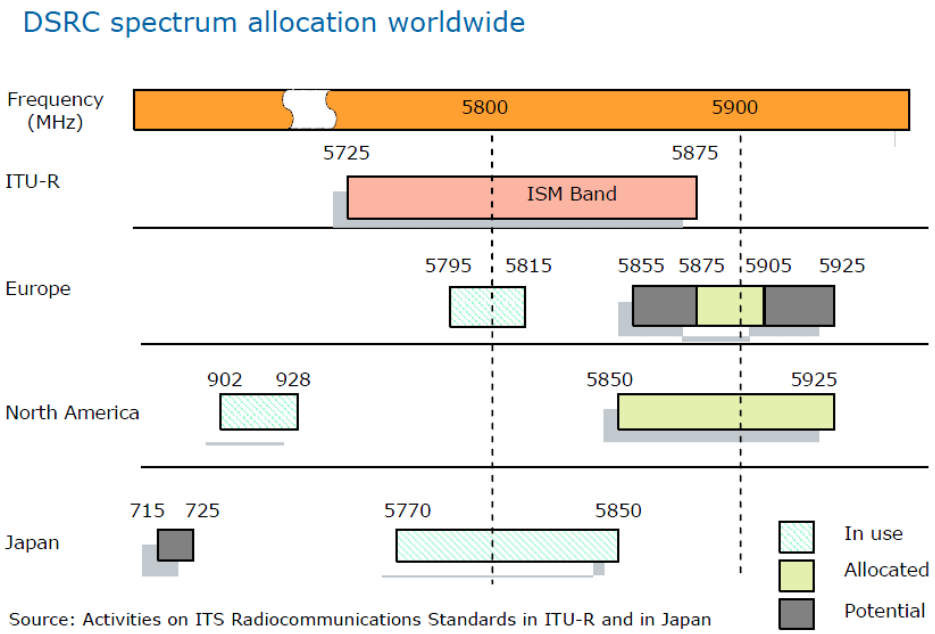

fig. 4 – Alocação de espectro para DSRC a nivel mundial (retirado de [7])

A performance espectral pode ser vista na figura 5, onde se pode ver que nos 5.9GHz o alcance pode chegar até aproximadamente 1Km.

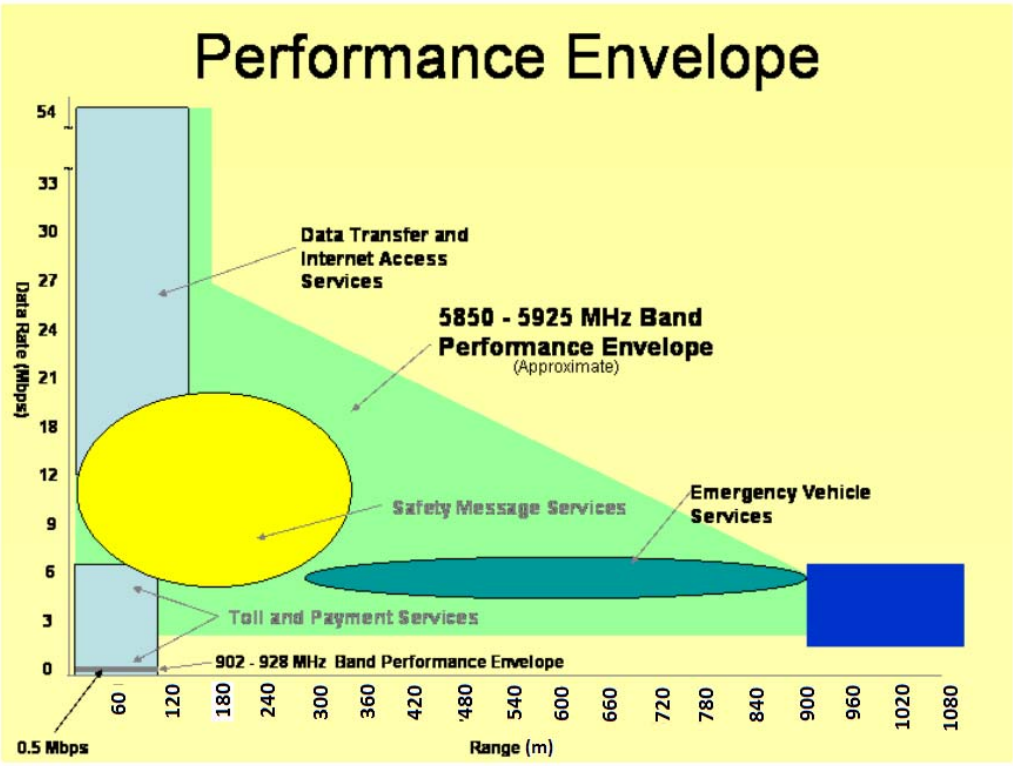

fig. 5 – Performance do DSRC (retirado de [8])

2.2.2 Aplicações, projectos em curso/finalizados

Em resumo enumero aqui algumas das muitas aplicações possíveis para o DSRC:

– **Segurança pública:**

• Assistente de aproximação de veículo de emergência.

• Prioridade do sinal de veículos de emergência.

• Aviso da condição da via pública.

• Aviso sobre as condições atmosféricas

- Formação de gelo na via.

- Avisos de *Aquaplaning* devido a chuvas

- Aviso de nevoeiro

• Aviso de ponte baixa.

• Aviso de zona de trabalho/obras.

• Aviso de colisão iminente.

• Assistente de velocidade numa curva.

• Infra-estrutura baseada num assistente de luz de travagem.

• Aviso de uma intersecção ou cruzamento.

• Impedimento de colisão numa auto - estrada.

• Aviso de colisão (veículo para veículo).

• Aviso de velocidade óptima.

• *Cruise control* adaptável.

• Informação de trânsito.

• Transferência de dados de trânsito entre veículos.

• Sinal de veículos de trânsito prioritários.

• Transmissão de vídeo de veículos de emergência.

• Projecção da via principal.

• Passeios desimpedidos.

• Inspecção do veículo.

• Diário do condutor.

– **Dados privados:**

• Controlo de acesso.

• Pagamento em condução.

• Pagamento em parques de estacionamento.

• Transferência de dados

  o Dados de diagnóstico.

  o Gravação de serviços de manutenção/reparação.

  o Actualização de programas computacionais do veículo.

  o Actualização de dados de música.

  o Carregamento de vídeos.

• Planeamento de rotas.

• Processamento de aluguer de veículos.

• Controlo rápido e único de operações de veículos comerciais.

• Monitorização do gasto de combustível

A empresa norte americana KAPSCH, anuncia em 2009 ter implementado o primeiro sistema de cobranças electrónica de portagens, baseado na tecnologia 5.9 GHz DSRC WAVE. Este sistema foi implementado no Porto de Hood River nos Estados Unidos. Este pode ser identificado como um marco no que diz respeito ao DSRC WAVE, pois é o primeiro sistema real implementado com

esta tecnologia, tendo sido verificado pelo FCC (*Federal Communications Commission*), as suas capacidades implementadas num sistema real com 5.9 DSRC WAVE [9].

A empresa DENSO desde 2008 anda em testes para conseguir comunicações entre veículos, estando a estudar a solução com UHF, e DSRC 5.8GHz, e anunciam no seu site que esperam ter em 2012 um produto comercializável para comunicações V2V [10].

Como se pode ver as aplicações existentes hoje em dia ainda não conseguiram ultrapassar totalmente os desafios colocados pela tecnologia, sendo bastante reduzida a oferta de produtos com o DSRC 5.9GHz, capazes de atingir as potencialidades referidas anteriormente. Daqui podemos ver a importância da investigação nesta área.

2.3 Rádios definidos por software

Em 1991 Joseph Mitola escreveu na sua tese de mestrado, quando se referia a rádios reprogramáveis ou reconfiguráveis, uma definição de *software defined radio* (SDR), que é hoje em dia considerada como a primeira.

"A software radio is a radio whose channel modulation waveforms are defined in software. That is, waveforms are generated as sampled digital signals, converted from digital to analog via a wideband Analog-to-Digital Converter (DAC) and then possibly upconverted from IF to RF. The receiver, similarly, employs a wideband ADC that captures all of the channels of the software radio node. The receiver then extracts, downconverts and demodulates the channel waveform using software on a general purpose processor." [11]

Este tipo de rádios pode ser definido muito resumidamente como uma única peça de hardware capaz de realizar diferentes funções em intervalos de tempo distintos, através de mudanças no software que o compõe. Assim com o mesmo hardware, através de alterações no

software, ter a possibilidade de funcionar com tecnologias e normas diferentes. Se pensarmos por exemplo num *smartphone* actual, vemos que estes incluem vários tipos de comunicações, como sejam, GSM, 3G, Wifi, Bluetooth etc, estes não podem ser considerados como SDR, pois para cada uma das tecnologias, tem um chip dedicado, cabendo apenas ao software a mudança para o que for pretendido pelo utilizador.

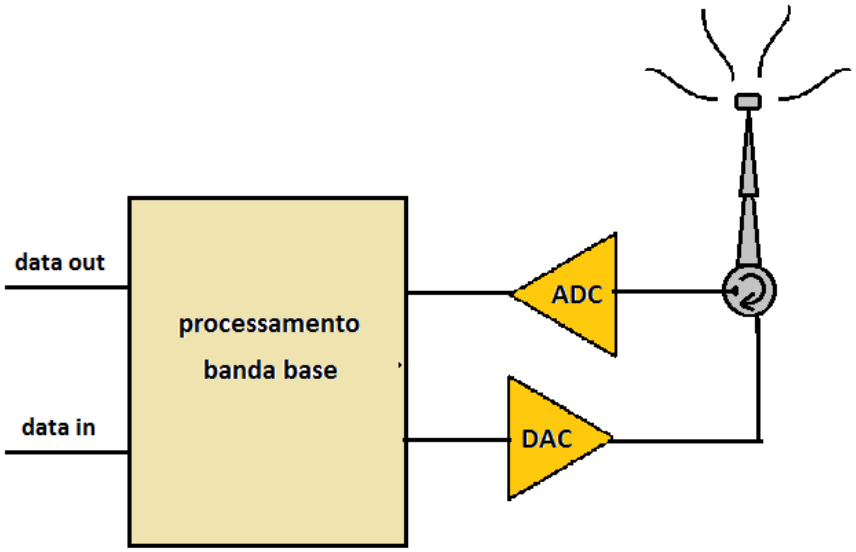

fig. 6 – Modelo de um SDR ideal.

Embora a definição concreta de o que é um SDR seja controversa e não reúna consenso no que concerne ao nível da reconfigurabilidade que este necessita de ter para que seja considerado como um, pois o facto de ser constituído por um DSP, FPGA ou ASIC, não significa necessariamente que estejamos na presença de um rádio definido por software. No entanto é aceite por todos que um rádio onde questões como modulação, correcção de erros, algum tipo de controlo sobre a parte de rádio frequência são feitos em software e podem ser reconfigurados facilmente, pode-se dizer que estamos claramente na presença de um SDR. Actualmente um típico exemplo de rádio definido por software, não é exactamente aquele que está representado na figura 6, pois nessa figura está ilustrada a situação ideal, em que da parte digital para a parte física, o meio, apenas existem conversores AD/DA, o que actualmente se implementa é o que está na figura 7.

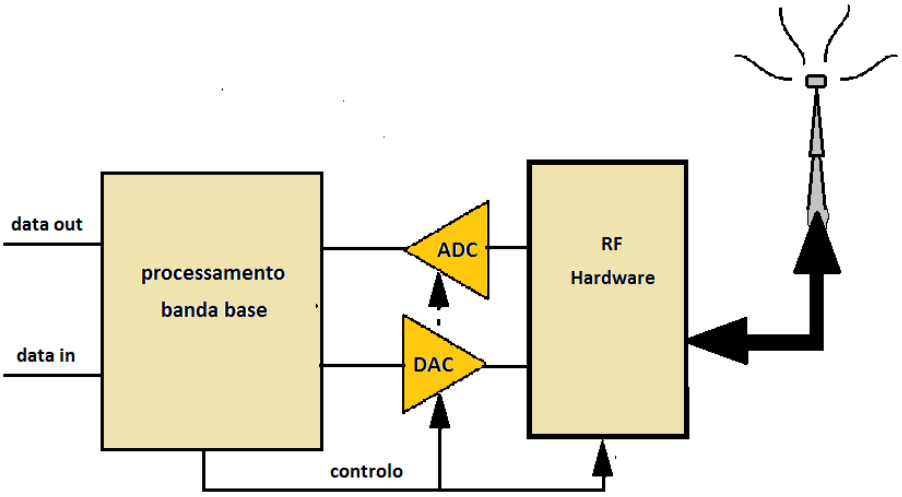

fig. 7 – modelo de um SDR actual

Devido a questões relacionadas com a tecnologia, ainda não é possível hoje em dia ter o exemplo da figura 6, isto devido a limitações nos componentes como DAC, ADC, antenas, que embora bastante evoluídos, não são ainda capazes de operar em gamas de frequência que vão desde alguns Hz ou MHz na banda base, ou na banda intermédia, até às dezenas de GHz, em que algumas tecnologias operam hoje em dia. Como tal a presença de um andar de rádio frequência que faça a ligação da parte de software, com o meio é uma presença habitual nos SDR que se implementam actualmente. No entanto nos últimos anos este tipo de rádios tem evoluído e "aproximado" da antena, isto é, os rádios actuais já tem a sua parte de software, a operar nas zonas de frequência intermédia, presentes nos rádios tradicionais.

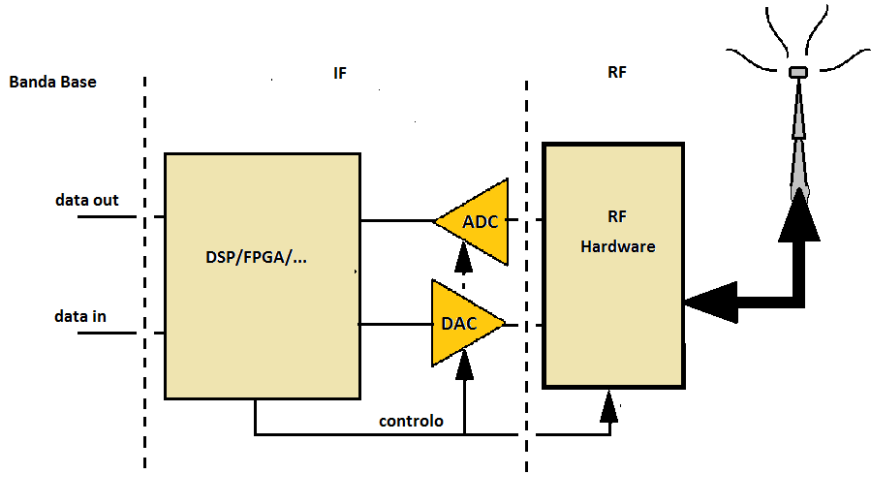

fig. 8 – Modelo de um SDR actual a actuar na frequência intermédia

2.3.1 Vantagens dos SDR

Com o avanço das telecomunicações, novos standards, evoluções e correcções de *buggs* dos antigos, e novas tecnologias estão constantemente a surgir. Este avanço nas tecnologias tem as suas claras vantagens, mas trás um grande inconveniente, é que faz com que os rádios das tecnologias anteriores fiquem obsoletos, pois o seu hardware já não é capaz de lidar com essas novas tecnologias que vão surgindo. Deste *handicap* dos rádios tradicionais, surge uma das grandes vantagens dos SDR, a reconfigurabilidade permite-lhes que, sem ter necessidade de alterar o hardware, possam funcionar em novas condições.

Outra das vantagens em comparação com os tradicionais, é a menor utilização de componentes analógicos, como misturadores, filtros, amplificadores, osciladores locais etc. Visto que estes são implementados através de técnicas de processamento de sinal na parte de software, ou mesmo de componentes que implementam sistemas completos, ou seja por exemplo como foi referido anteriormente, num *smartphone* os diferentes tipos de comunicações possíveis que existem nestes telemóveis têm um circuito dedicado para cada uma delas, nos SDR estarão todos integrados no mesmo circuito. Isto trás vantagens ao nível do tamanho e do consumo de potência que são factores de extrema importância nos equipamentos móveis principalmente. O menor número de componentes pode por si só, trazer uma redução nos custos, embora os componentes de hardware como FPGAs e DSPs, terem ainda valores elevados, estes valores são compensados, por um menor custo na fase de testes e de desenvolvimento dos equipamentos, pois muitos dos possíveis defeitos podem ser corrigidos sem necessidade de novos componentes. Devido à reconfigurabilidade, a implementação de novas tecnologias, ou de evoluções das anteriores, passa também a ser uma possibilidade com estes rádios. Outra grande potencialidade é a de poderem dois equipamentos com tecnologias diferentes poderem comunicar entre si, por exemplo, um fax equipado com *bluetooth*, poderá enviar um fax para um computador que tenha um SDR implementado que contemple *bluetooth*. Se pensarmos em termos geográficos, vemos outra das vantagens, pois de pais para pais é natural que as tecnologias usadas sejam diferentes, e portanto com os aparelhos tradicionais para que se poder ter conectividade em qualquer região, implica ter vários dispositivos. Com um SDR será possível com o mesmo dispositivo ter conectividade global. Graças a estas vantagens estes rádios são apelidados de “rádios à prova de futuro”[12][13].

2.3.2 Rádios Cognitivos e Análise de Espectro

Com as potencialidades dos SDR, novos paradigmas nos rádios têm sido pensados, e se na secção anterior os SDR foram chamados de rádios à prova de futuro, estes são o futuro dos rádios do futuro.

Assim como não existe consenso nos SDR quanto ao nível de software que os compõem, também não existe consenso nos *Cognitive Radios* (CR) ou em português rádios cognitivos, no que diz respeito ao nível de inteligência que estes têm de ter. Surgem então várias definições, por exemplo, Joseph Mitola define em 1999 rádios cognitivos como:

"A radio that employs model based reasoning to achieve a specified level of competence in radio-related domains" [14].

Ou a definição de Simon Hawkin's

"An ambient-aware, intelligent radio which learn from its surroundings and adapt itself to:

- *Highly reliable communication, anywhere, anytime;*
- *Efficient use of Radio Spectrum;"* [15].

São algumas das definições que têm sido dadas.

Basicamente os CR pretendem ir muito mais além no que diz respeito ao tipo de comunicações que utilizam, pois estes mais do que poder utilizar vários tipos de standards nas suas comunicações, os *CR* tem o objectivo de analisar o meio em que vão transmitir, e tomar decisões como: que modulação utilizar, que nível de potência, canal e frequência de transmissão entre outros. Nesta linha raciocínio surge o conceito de rádios completamente inteligentes que são actualmente a utopia no que aos rádios diz respeito, serão uma espécie de rádios cognitivos que ainda vão mais além, pretende-se que sejam capazes até de formar os seus próprios protocolos de comunicação, que surgem de um "acordo" entre o transmissor e o receptor, de forma completamente autónoma e transparentepara o utilizador.

Para que seja possível atingir estes objectivos uma análise do espectro tem de ser obrigatoriamente feita. Nesta análise o rádio tem de ver onde os rádios vizinhos estão a transmitir de modo a que não se sobreponham transmissões de rádios diferentes. Tendo em conta o a gama de espectro em que pode transmitir, o rádio pode decidir qual a mais adequada para a transmissão em causa. A análise de espectro consiste em ver onde existem espaços livres no espectro para se poder transmitir, e o que se espera conseguir com os CR, é algo semelhante ao que está representado na figura 9.

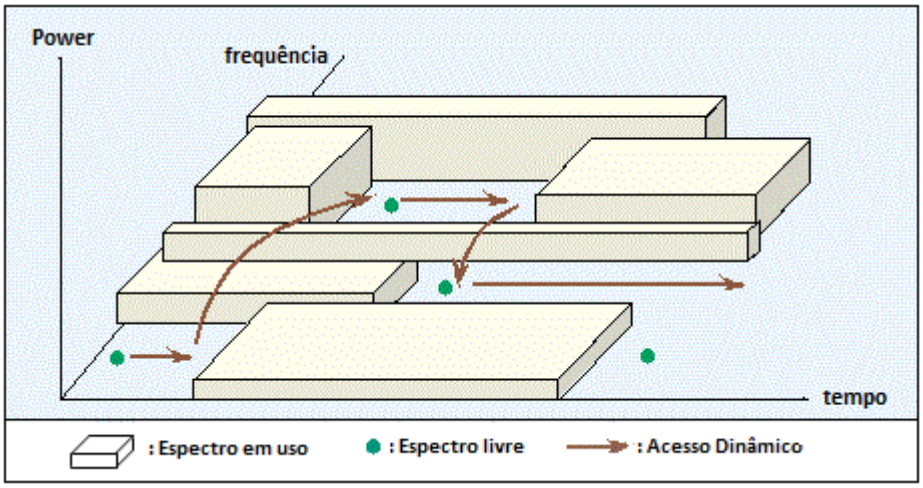

fig. 9 – Análise e utilização inteligente do espectro (retirado de [16])

Ou seja o rádio deve ver onde é que existe espaço livre ao longo do tempo para poder transmitir, e consoante os espaços vazios pode até dar saltos para frequências livres, como está exemplificado na figura, e deste modo fazer uma utilização do espectro de uma forma muito eficiente. Deste modo um CR não precisaria no limite de ter qualquer banda de frequência alocada previamente com a entidade que regula o espectro (em Portugal a ANACOM), mas esta questão levanta questões politicas e financeiras, pois actualmente a utilização do espectro é cobrada pelos seus detentores sendo este uma forma de receitas dos países. Estas questões que se levantam serão possivelmente tão ou mais difíceis de ultrapassar que os desafios da académicos que a tecnologia terá para oferecer!

2.3.3 Ligação com o projecto HEADWAY

No âmbito do projecto que dá origem a esta tese, pretende-se implementar a camada física, bem como a camada MAC, numa FPGA da XILINX, sendo estas duas camadas as camadas onde a definição de SDR coloca as suas modificações em relação aos rádios tradicionais. No que diz respeito à camada física, esta tem implementado em FPGA, as questões de correcção de erros, mapeamento e modulação OFDM. Cabe também à FPGA, através de um processador embutido (*Microblaze*), fazer o controlo das camadas que funcionam como periféricos para o processador,

bem como fazer o controlo do módulo de RF e dos conversores ADC e DAC, como se pode ver esquematizado na figura 10.

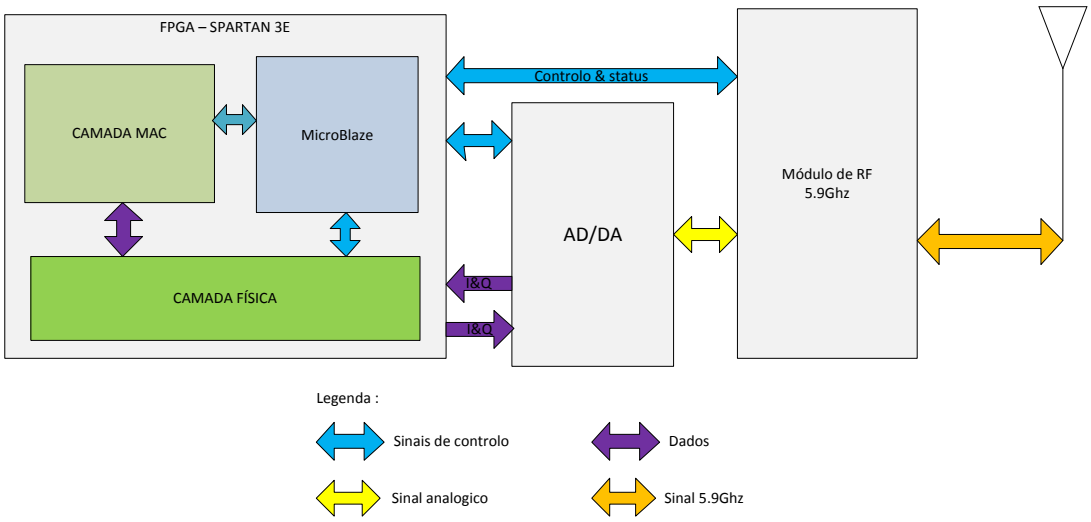

fig. 10 – Esquema do rádio do projecto HEADWAY

3 OFDM em 802.11p

Os requisitos que os actuais sistemas de comunicação exigem, requisitos como taxas de transmissão elevadas garantidas mesmo em condições de transmissão bastante adversas, requerem modulações por vezes complexas para poder cumprir com os requisitos. Uma das modulações que permite cumprir as necessidades actuais, é o OFDM. Este está presente em várias tecnologias actuais, em particular no IEEE 802.11p.

3.1 OFDM

OFDM (*Orthogonal Frequency Division Multiplexing*), é uma técnica de modulação que consiste na divisão da frequência, onde múltiplos sinais são enviados em frequências diferentes. Esta técnica é uma evolução da técnica de FDM (*Frequancy Division Multiplexing*), onde os vários sinais são enviados em diferentes portadoras com um intervalo de guarda entre estas. No OFDM recorrendo a portadoras ortogonais, estas não necessitam de intervalos de guarda, podendo mesmo ser sobrepostas e mesmo assim garantir a não interferência entre elas, providenciando desta maneira uma alta eficiência espectral e consequentemente para o mesmo espectro, taxas de transmissão muito mais elevadas.

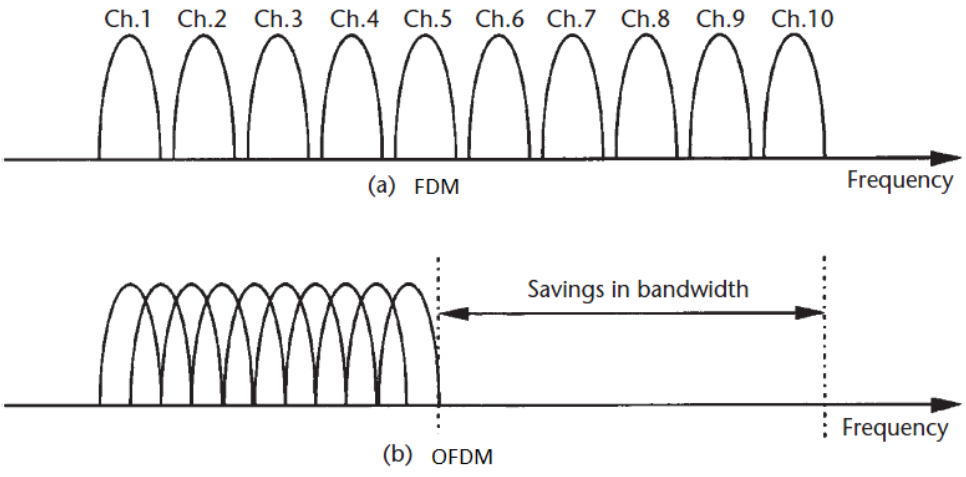

fig. 11 – Comparação da largura de banda do sistema FDM tradicional com o OFDM (retirado de [17])

Duas analogias pictóricas normalmente usadas para comparar FDM e OFDM, são a da torneira e do chuveiro, ou seja se imaginarmos que o FDM como uma torneira, podemos dizer que o OFDM é o chuveiro, qual a vantagem disto? Se taparmos com o polegar o único canal que existe na torneira, tem como consequência o corte inteiro de todo o fluxo de água que dela sai, enquanto que no chuveiro é muito mais difícil tapar todos os fios individuais que dele saem (correspondendo estes às portadoras ortogonais), ou seja mesmo tapando alguns deles com o polegar o fluxo de água nunca é completamente interrompido. (figura 12 (a))

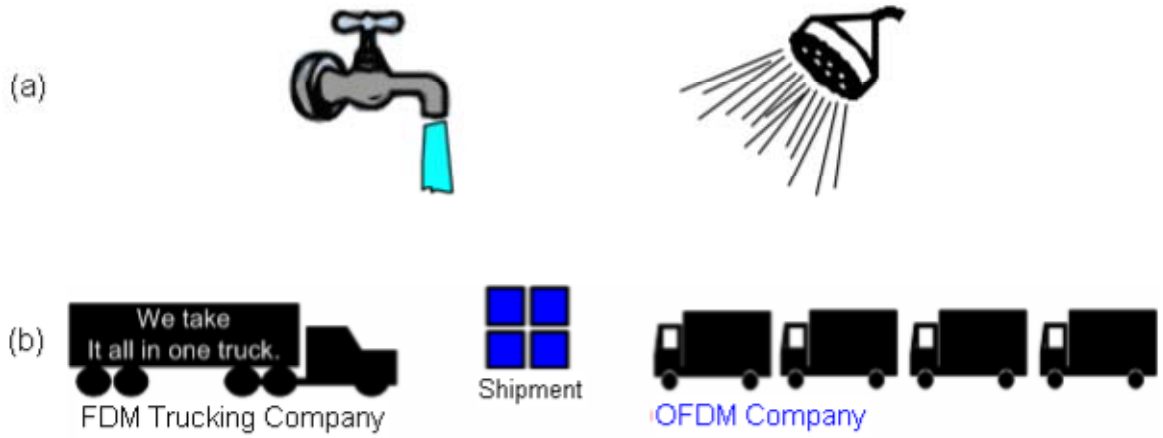

fig. 12 –Analogias FDM vs OFDM (retirado de [17])

A outra muito usada é como se pode ver na alínea b) da figura 12, a comparação com camiões de transporte, em que no FDM, ou seja o camião grande, toda a carga é transportada por esse mesmo camião, e portanto se houver algum problema durante o transporte toda a carga não irá chegar ao destino, enquanto que na outra situação da carga dividida por quatro pequenos camiões, se houver um problema com um dos camiões, três quartos da carga irá chegar ao destino. Destas analogias podemos ver que uma das grandes vantagens desta técnica é a sua robustez contra as interferências de banda estreita e desvanecimento. Num sistema de portadora única o desvanecimento ou interferência na banda, pode levar à falha completa do sistema, enquanto que num sistema multi-portadora como o OFDM, apenas algumas das suas portadoras são afectadas. Este problema pode ser contornado com a introdução de códigos correctores de erro e *interleaving* da bit-*stream* [17].

Esta técnica foi apresentada pela primeira vez na década de 60, sendo patenteada em 1970 nos Estados Unidos da América, sendo pela primeira vez implementado em sistemas de comunicações sem fio na década de 90 [18].

Devido à complexidade de implementação até então, apenas na década de 90 sistemas OFDM começaram a ser realmente implementados, tendo sido adoptado pelos sistemas cabelados ADSL (*Asynchronous Digital Subscriber Line*), e HDSL (*High-bit-rate Digital Subscriber Line*). E nos

sistemas sem fios Europeus de DAB (*Digital Audio Broadcasting*) e DVB (*Digital Video Broadcasting*). Em 1999 foi adoptado também pelo IEEE para o IEEE 802.11a,g e actualmente na emenda p do 802.11 que está a ser alvo de estudo neste documento [19].

3.1.1 Implementação

Nos sistemas tradicionais, os símbolos são enviados em sequência apenas por uma portadora, onde o espectro de cada símbolo ocupa toda a largura de banda disponível para transmissão.

A técnica OFDM é baseada, como foi dito anteriormente, em portadoras ortogonais e em sobreposição espectral das portadoras, o que resulta numa compressão do espectro em comparação com os sistemas FDM tradicionais. Por outras palavras esta técnica consiste no envio de dados paralelos em várias portadoras. Estas portadoras são normalmente moduladas em PSK (*Phase Shift Keying*) ou em QAM (*Quadrature Amplitude Modulation*), estando os ritmos de transmissão relacionados com o tipo de modulação escolhida e com o número de portadoras.

Para melhor se entender este conceito, vejamos o caso de uma portadora modulada com um pulso rectangular, e o seu correspondente espectro da forma de sinc(fT). A intersecção deste espectro da forma de sinc com zero acontece nos pontos múltiplos de 1/T, em que T é o período do pulso rectangular (símbolo), e sendo a frequência central correspondente ao zero do eixo horizontal, a frequência real da portadora.

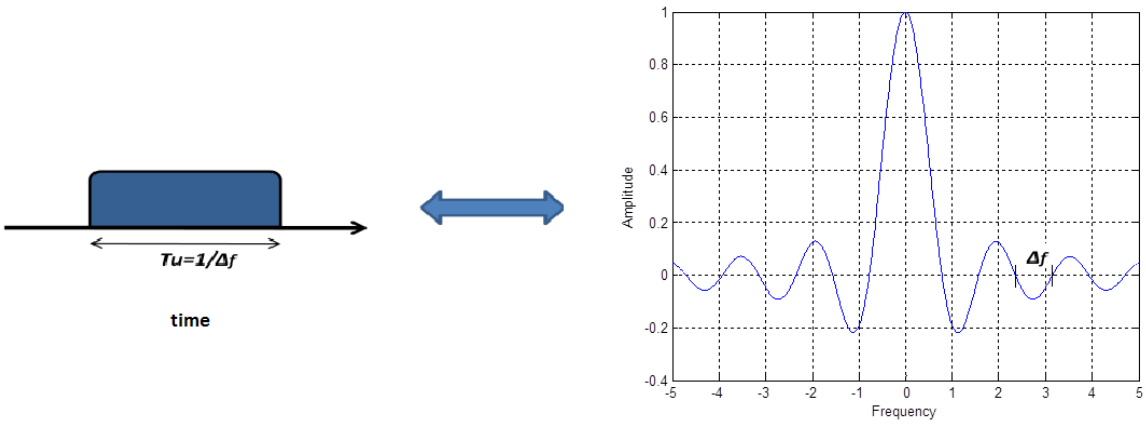

fig. 13 – Representação no tempo e frequência de um pulso rectangular

Se considerarmos em vez de uma portadora, considerarmos várias, podemos obter um espectro de um FDM tradicional, onde não existe qualquer relação entre as portadoras, embora se tenha de considerar um intervalo de guarda entre as frequências de modo a evitar interferência entre

canais adjacentes. Ou podemos ter o Espectro de um sinal OFDM, se considerarmos frequências das portadoras, centradas nas frequências múltiplas de 1/T, como podemos ver na figura 14.

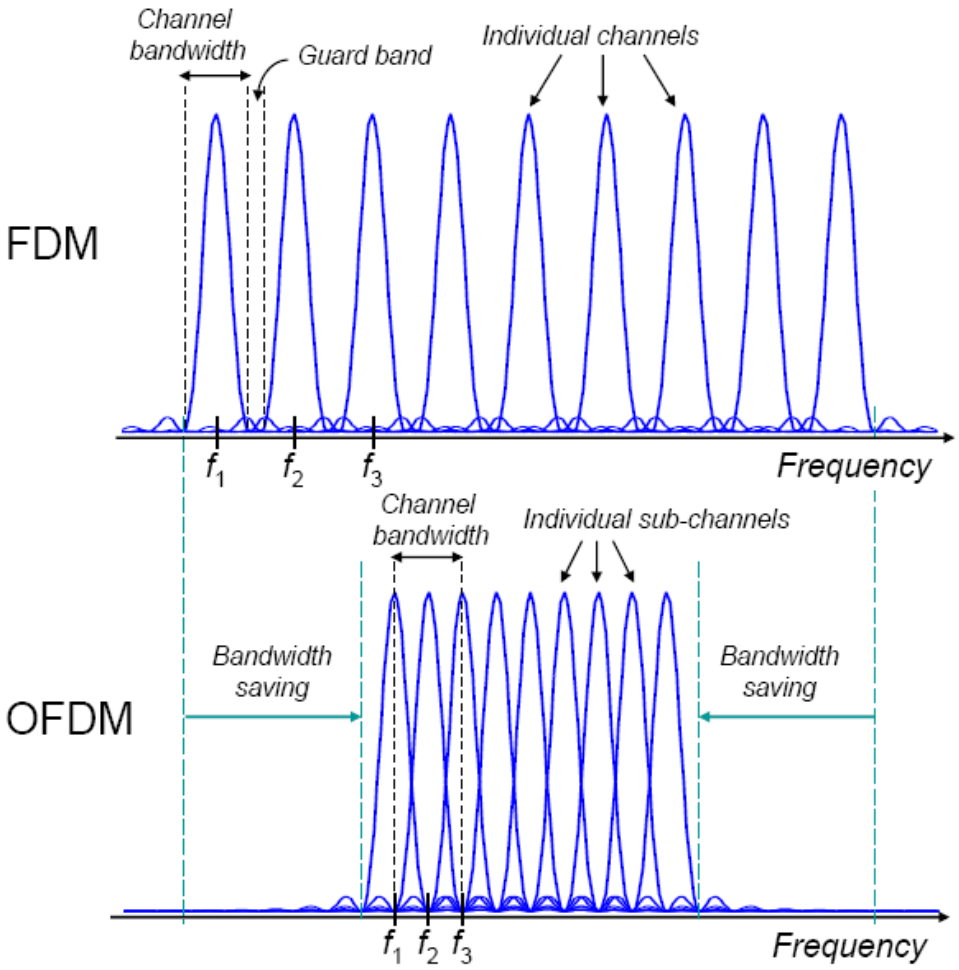

fig. 14 – Espectro FDM convencional e OFDM

Apesar de no espectro OFDM os espectros estarem claramente sobrepostos, estes não causam interferência entre si devido à ortogonalidade das portadoras, ou seja o produto interno entre as portadoras é zero [18].

3.1.2 Geração das portadoras usando IFFT

Como foi dito anteriormente as portadoras são moduladas em PSK ou QAM. Pegando por exemplo no caso de modulação em QAM, considerando d_i os símbolos complexos QAM, N_s o número de portadoras, T a duração do símbolo e f_c a frequência da portadora, e considerando o símbolo OFDM que começa em $t = t_s$, temos:

$$s(t) = \sum_{i=-\frac{Ns}{2}}^{\frac{Ns}{2}-1} d_{i+\frac{Ns}{2}} e^{j\frac{2\pi i}{T}(t-t_s)} \quad , \quad t_s \leq t \leq t_s + T$$

$$s(t) = 0 \quad , \qquad t < t_s \;\; e \;\; t > t_s$$

Assim desta forma um possível modulador QAM-OFDM poderia ser o da figura 15

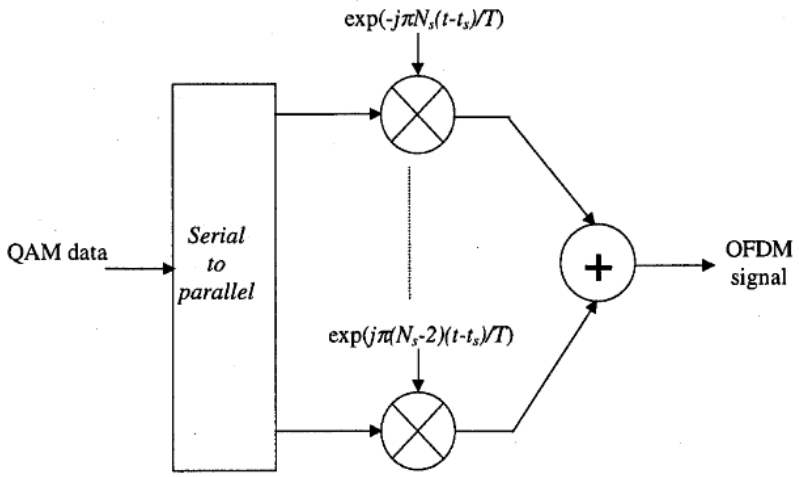

fig. 15 – Modulador QAM-OFDM (retirado de [18])

Mas se repararmos na equação (3.1), podemos identificar que se trata da equação de uma transformada de fourier (DFT), de N_s símbolos QAM.

Ou seja na prática as portadoras do sistema OFDM podem ser conseguidas recorrendo ao algoritmo computacionalmente eficiente que calcula a IDFT, a IFFT. Esta solução de implementar o sistema através de IDFT foi apresentada pela primeira vez em 1971 por *Weinstein e Ebert*.

Esta solução traz grandes vantagens, pois recorrendo à IFFT, na transmissão, e a FFT na recepção, estes sistemas tornam-se ideais para a implementação em FPGAs e DSPs, e os fabricantes destes equipamentos fornecem, na forma de *IP-CORES* que realizam FFT/IFFT, muitas vezes oferecem mesmo versões gratuitas destes *IP-CORES*.

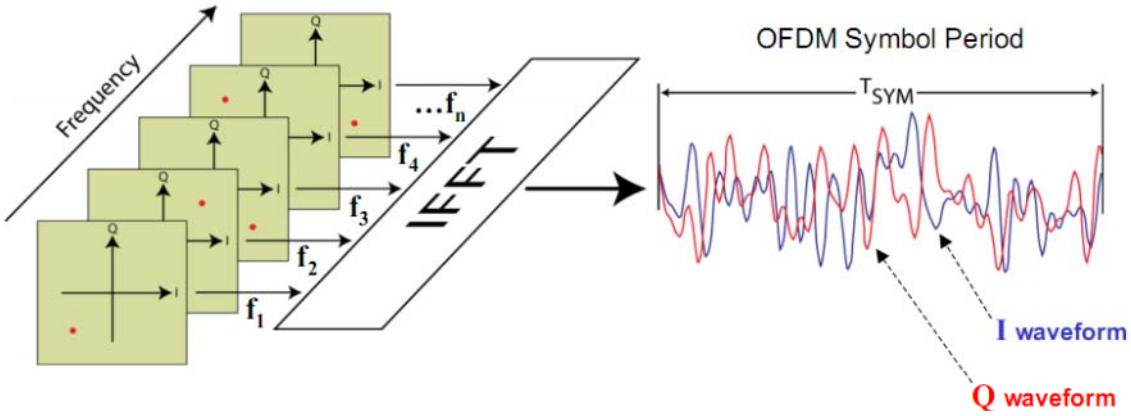

fig. 16 –Implementação de portadoras com IFFT(retirado de [20])

Usar esta técnica tem como consequência que o receptor tenha de sincronizar muito bem no tempo e na frequência, de modo a que a ortogonalidade das portadoras seja garantida. Para ajudar na tarefa de sincronização, é usada a técnica de introdução de portadoras piloto, que são divididas entre as portadoras de dados. Estas portadoras servem como um padrão conhecido pelo receptor para ajudá-lo na sincronização.

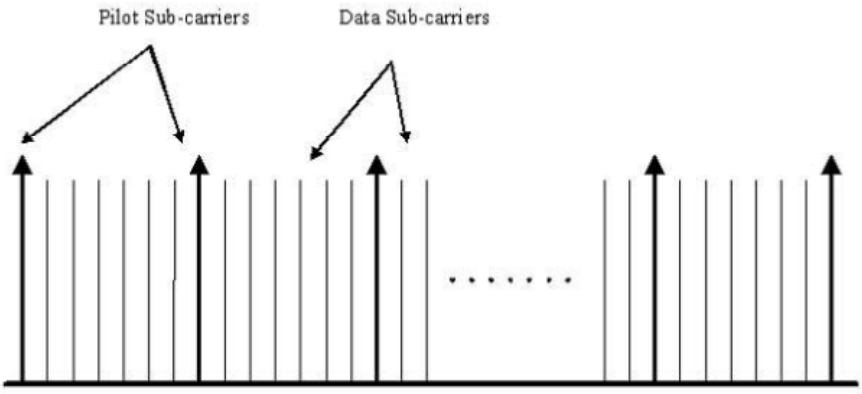

fig. 17 – Portadoras piloto

3.1.3 O problema do multi-percurso, e o prefixo cíclico

Numa transmissão rádio, o sinal electromagnético que chega ao receptor é para além do sinal em linha de vista (LoS) entre o emissor e receptor, é a soma de várias reflexões que ocorrem durante a transmissão. O sinal que o receptor recebe é a soma desses vários sinais, com diferentes ganhos e atrasos sofridos nos diferentes percursos.

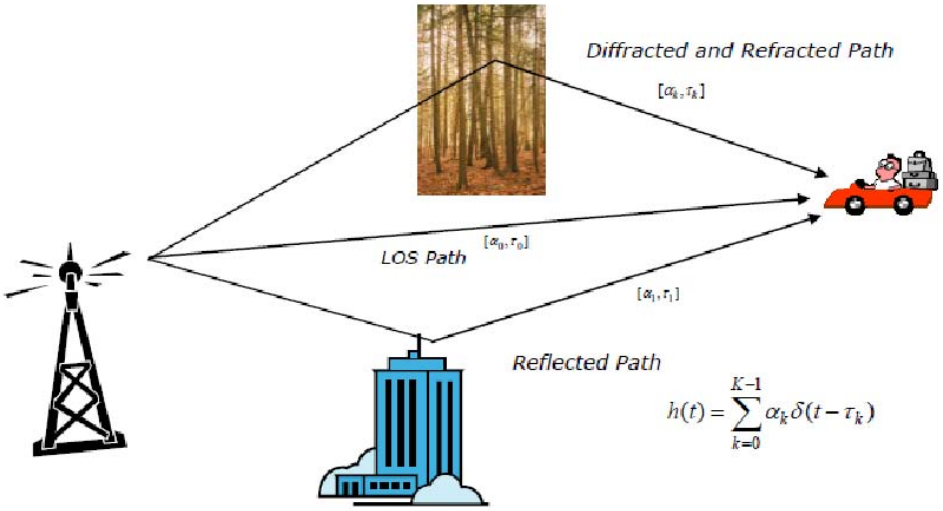

fig. 18 – Exemplo de canal de comunicação com multi-percurso

Outro problema a ter em conta é o desvanecimento, podendo cada portadora sofrer diferentes atenuações. Desvanecimento pode ser definido como as diferentes atenuações que uma

portadora modulada sofre ao longo do meio de transmissão, atenuação esta que pode variar com o tempo, o local e com a própria frequência da portadora. Os principais efeitos causados pelo canal multi-percurso são o desvanecimento selectivo de frequência e a interferência entre símbolos (*ISI*). Na figura 19 podemos ver um exemplo de desvanecimento profundo de algumas frequências. Se no caso do FDM desvanecimentos profundos levam à inevitável perda de um símbolo, no OFDM, apenas algumas portadoras são afectadas, e consequentemente só se perdem alguns bits e não todo o símbolo, e deste modo recorrendo à codificação e ao *interleaving* antes de se fazer o mapeamento QAM ou PSK, o efeito do desvanecimento que levou a perda de algumas portadoras é minimizado. O desvanecimento selectivo da frequência pode ser evitado aumentado o número de portadoras.

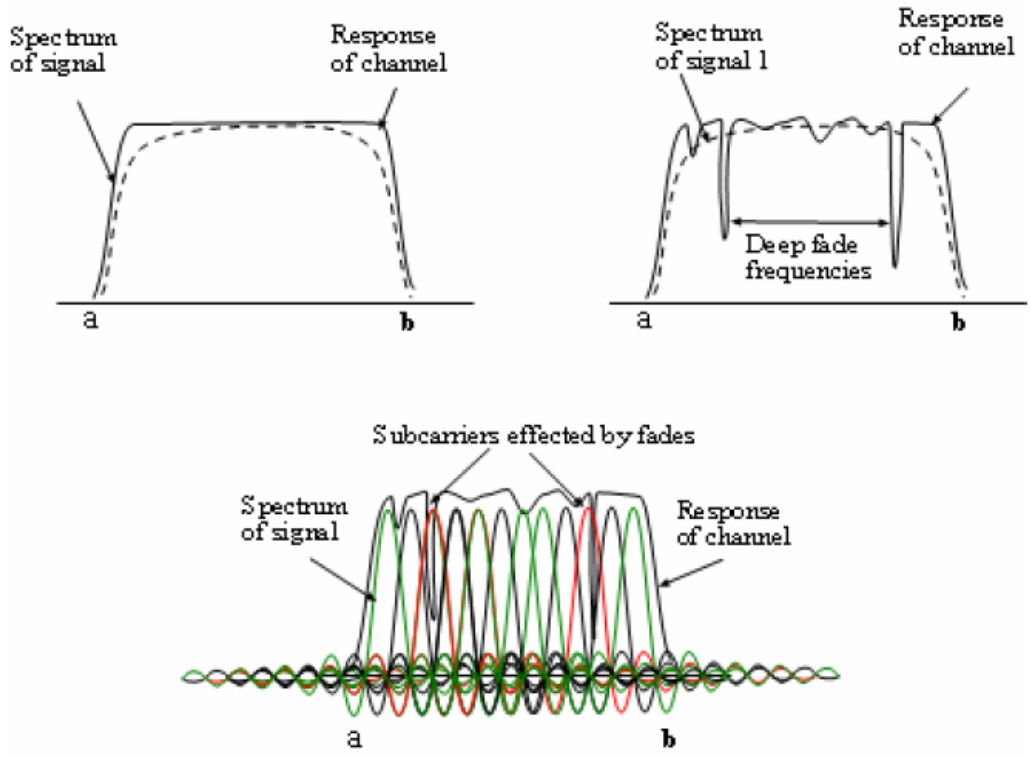

fig. 19 – Espectro de FDM E OFDM, e resposta ao desvanecimento do canal (retirado de [17])

Outra das consequências do multi-percurso é, como já foi dito, a interferência entre símbolos (*ISI*). A interferência pode ser vista como dois carros que vão seguidos na mesma estrada, num dia chuvoso, e o carro que passa em primeiro lugar salpica água de uma possa no chão, para que o segundo carro não seja atingido com o salpico que o primeiro originou, este pode dar um intervalo maior de distância entre eles. Se considerarmos o salpico originado pelo primeiro carro como a interferência causada por um símbolo, podemos calcular também para símbolos

consecutivos, a distância para que o segundo não seja afectado pelo primeiro [17].

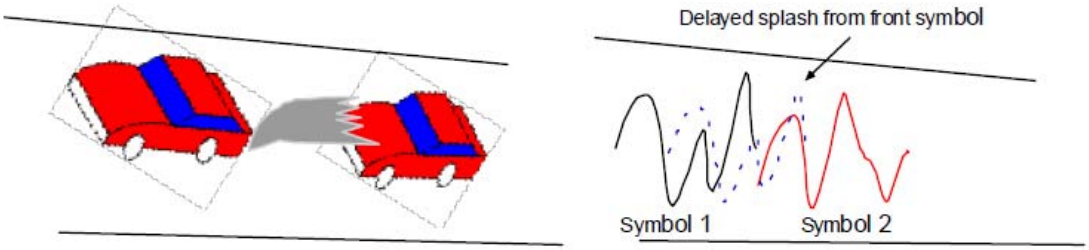

fig. 20 – Exemplo de interferência entre símbolos com carros (retirado de [17])

Para evitar a *ISI* a duração dos símbolos OFDM é aumentada, sendo acrescentado no inicio de cada símbolo um intervalo de guarda, conhecido por prefixo cíclico. Este prefixo cíclico é basicamente feito através de uma copia da parte final do símbolo, colocada no inicio do símbolo. Este intervalo não pode ser apenas um espaço vazio pois dessa forma perder-se-ia a ortogonalidade nas portadoras, e passava a existir interferência entre as portadoras.

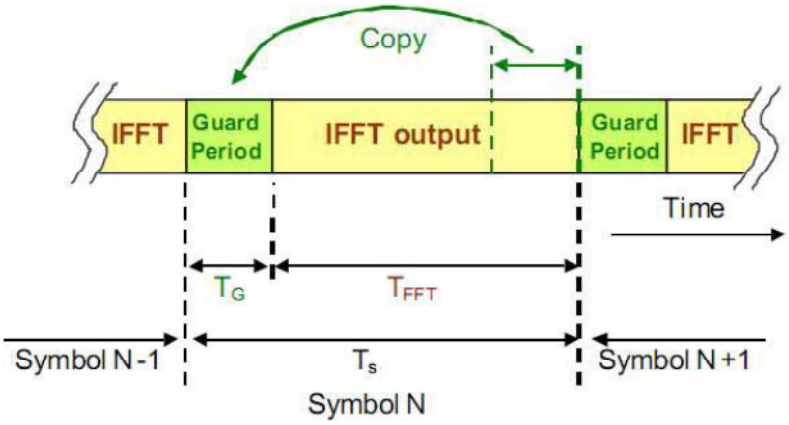

fig. 21 – Prefixo cíclico (retirado de [21])

A escolha do tamanho do prefixo cíclico deve ser um compromisso entre o evitar a interferência entre símbolos, e a relação sinal ruído (*SNR*), pois o prefixo cíclico não transporta informação relevante e introduz ruído, o que vai fazer baixar a *SNR*, por outro lado se não tiver o comprimento adequado ao atraso introduzido pelo canal, podemos ter *ISI*.

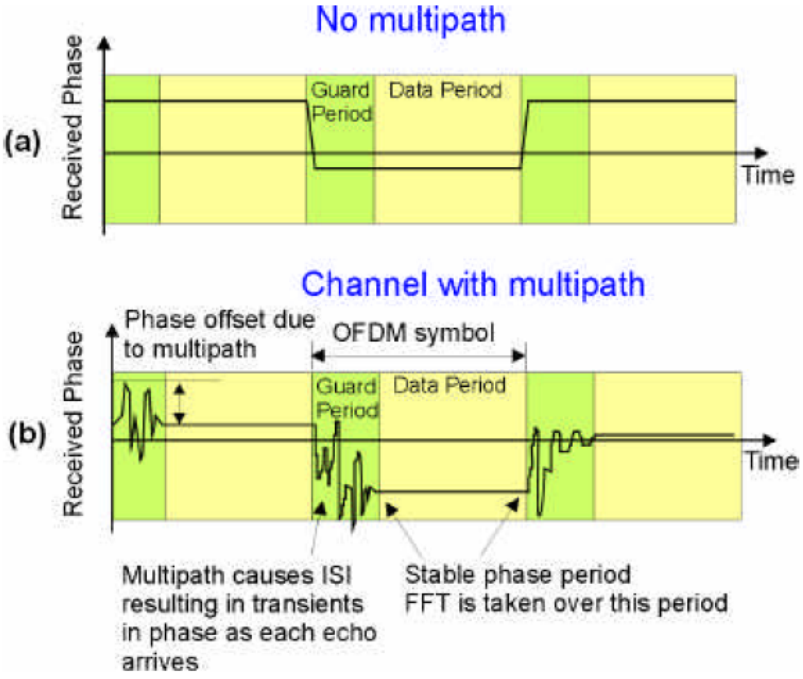

fig. 22 – Interferência entre símbolos no intervalo de guarda(retirado de [21])

Na figura 22 podemos ver como o intervalo de guarda, funciona como protecção contra interferência entre símbolos em canais multi-percurso.

No receptor para o processo de passagem do domínio do tempo para o da frequência, é realizado como já foi dito a FFT, tendo esta de ter o mesmo tamanho que a IFFT do emissor de modo a tirar partido das portadoras ortogonais, por isso antes da operação de FFT no receptor o prefixo cíclico tem de ser retirado, e assim as zonas onde está presente a interferência entre símbolos é eliminada.

3.1.4 Windowing

No OFDM as transições entre os símbolos transmitidos podem ter mudanças bruscas de amplitude, o que tem como consequência, que os lobos laterais no espectro de frequência, sejam bastante grandes. A potência elevada destes lobos pode cair em canais adjacentes, o que vai fazer com que erros aconteçam, e consequentemente a bit-error-rate (*BER*) aumente. A supressão de emissões fora da banda é bastante importante e exigida pelo regulador do espectro (em Portugal a ANACOM). Podemos ver na figura 23 que o aumento do número de portadoras faz com que o decaimento fora da banda seja muito mais elevado. No entanto muitas vezes nestes sistemas é aplicada uma janela temporal para garantir o decaimento necessário para as transmissões fora da banda. Para confinar o sinal no espectro é normalmente utilizado nos sistemas QAM o co-seno

elevado, e podemos ver na figura 24 que um pequeno factor β = 2.5% já faz uma grande diferença [18].

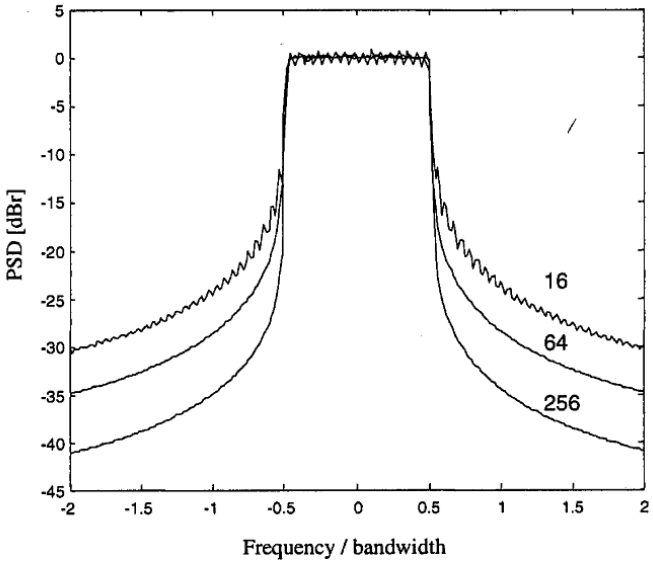

fig. 23 – Espectro OFDM com 16, 64 e 256 portadoras(retirado de [18])

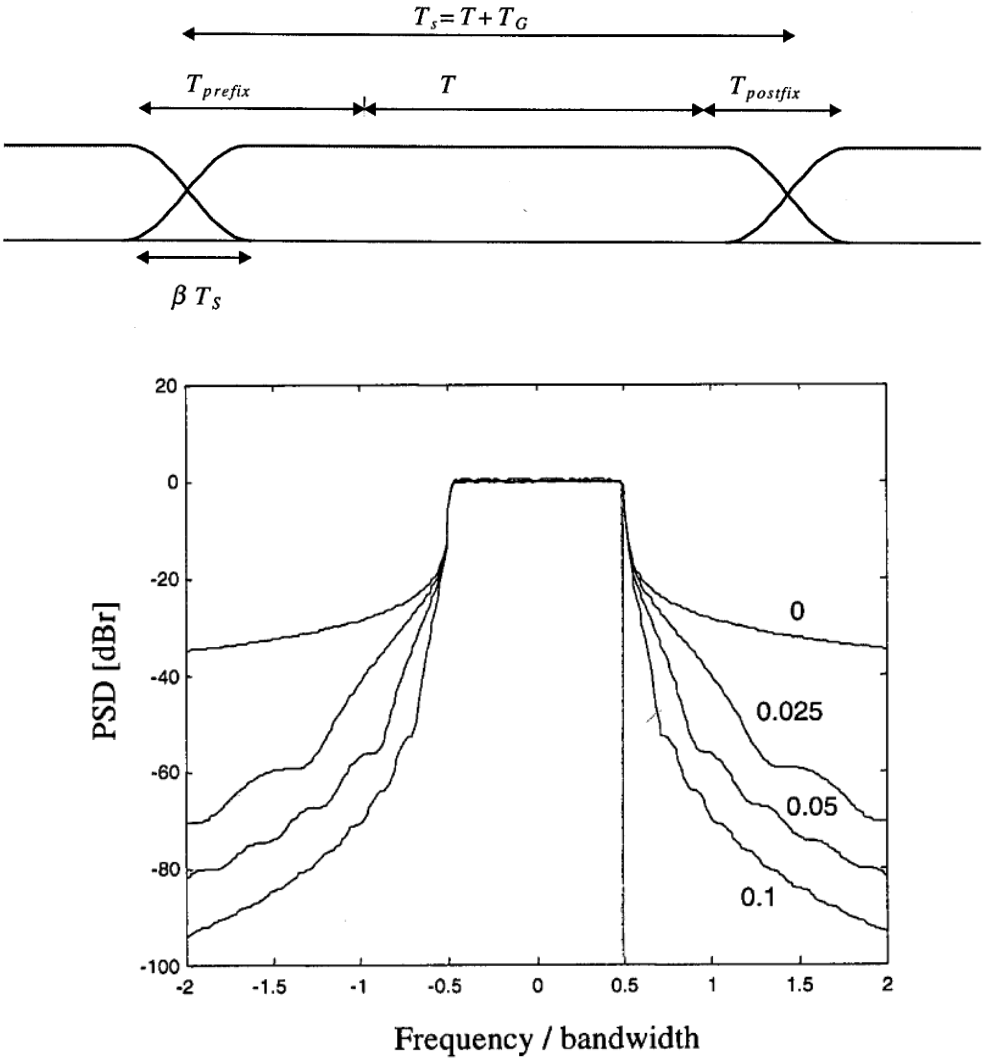

fig. 24 – Transição entre símbolos com coseno elevado (retirado de [18])

3.1.5 O sistema OFDM

Juntando tudo o que foi dito anteriormente sobre como funciona o ODFM, chegamos à configuração típica deste sistema na figura 25.

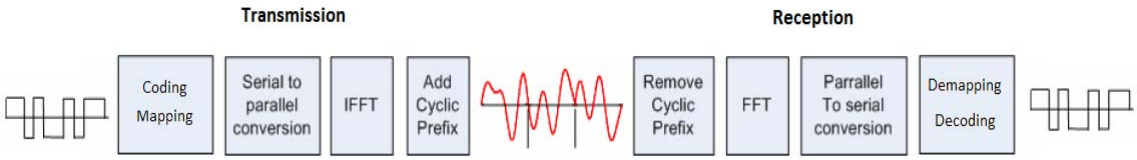

fig. 25 – Esquema de blocos do sistema OFDM completo (retirado de[18])

O sistema é constituído pela codificação, que tem como um dos objectivos, reduzir o impacto do canal com multi-percurso, nas portadoras. Depois da codificação é feito o mapeamento em QAM ou PSK. De seguida paralelizam-se os dados de forma a realizar a IFFT, de modo a obter as portadoras ortogonais no tempo. Depois da IFFT, é adicionado o prefixo cíclico, para evitar a interferência entre símbolos. Na recepção temos as operações inversas, ou seja, começasse por retirar o prefixo cíclico incluído na transmissão, de seguida a operação inversa da IFFT, a FFT é realizada para voltar ao domínio da frequência e de seguida fazer o desmapeamento das portadoras, e descodificação que foi introduzida, e finalmente temos os dados transmitidos.

3.1.6 Vantagens e Desvantagens do OFDM

Neste ponto em jeito de conclusão apresento as principais vantagens e desvantagens do OFDM.

Vantagens:

- Elevada eficiência espectral - O sinal pode ser moldado de modo a usar da forma mais eficiente possível a largura de banda disponível, atendendo a parâmetros como a modulação, o numero de portadoras e as condições do canal.

- Elevada eficiência em ambientes multi-percurso - aumentando a duração dos símbolos com a inserção do prefixo cíclico, que evita a interferência entre símbolos.

- Baixa complexidade dos receptores. - Devido à não existência de interferência entre as portadoras (*ICI*), e entre símbolos (*ISI*).

- Diferentes esquemas de modulação e codificação podem ser usados nas portadoras o que resolve o problema de desvanecimentos selectivos de frequência ou interferências de banda estreita.

Desvantagens:

- Necessidade de sincronização precisa no tempo e na frequência.
- Maior sensibilidade ao efeito *doppler* que em sistemas de portadora única.
- Sensibilidade a desvios de frequência e ruído de fase causadas por imperfeições dos osciladores, tanto na transmissão como recepção, que levam a interferência entre portadoras (*ICI*).
- Elevado PAPR (*Peak to Average Power Ratio*).

A reter os pontos a ter em conta no desenho de um sistema ODFM são;

- Espaçamento das portadoras e a duração do símbolo OFDM.
- O Número de portadoras.
- Duração do prefixo cíclico.

Para diminuir o elevado PAPR do OFDM, podem ser usadas várias técnicas, como *cliping*, mapeamento e codificação selectivos entre outros. Estas técnicas devem ser escolhidas tendo em atenção os requerimentos do sistema e tendo em conta o compromisso entre PAPR, simplicidade do sistema e taxa de transmissão [19].

3.2 Camada física 802.11p

A camada física de um protocolo, habitualmente chamada de *PHY,* é a camada de mais baixo nível do modelo OSI, ou seja a camada que faz a ligação entre meio, e a camada de controlo de acesso ao meio (MAC). Deste ponto de vista a sua função da *PHY* é atender aos pedidos de transmissão/recepção de dados feitos pela camada MAC. Visto que esta é camada de mais baixo nível, aqui a informação deixa de ser tratada como pacotes em que se juntam cabeçalhos, aqui é

bit a bit que se processam os dados. É nesta camada que são introduzidos/retirados desde, códigos de linha, códigos correctores de erros, é aqui que é feita a modulação/desmodulação, a conversão dos sinais digitais em sinais analógicos para serem transmitidos no meio, na frequência, e energia correcta. É também nesta camada que são definidas as características do hardware, conectores, dimensões, e todas as características eléctricas para se ligar ao meio.

Esta camada está fortemente dependente da tecnologia de comunicações que utiliza, ao contrário das camadas superiores, visto que tecnologias, frequências, potências de comunicação diferentes requerem hardware em consonância com as mesmas, o que pode ser resolvido recorrendo ao conceito SDR, e circuitos reconfiguráveis como será o caso neste trabalho.

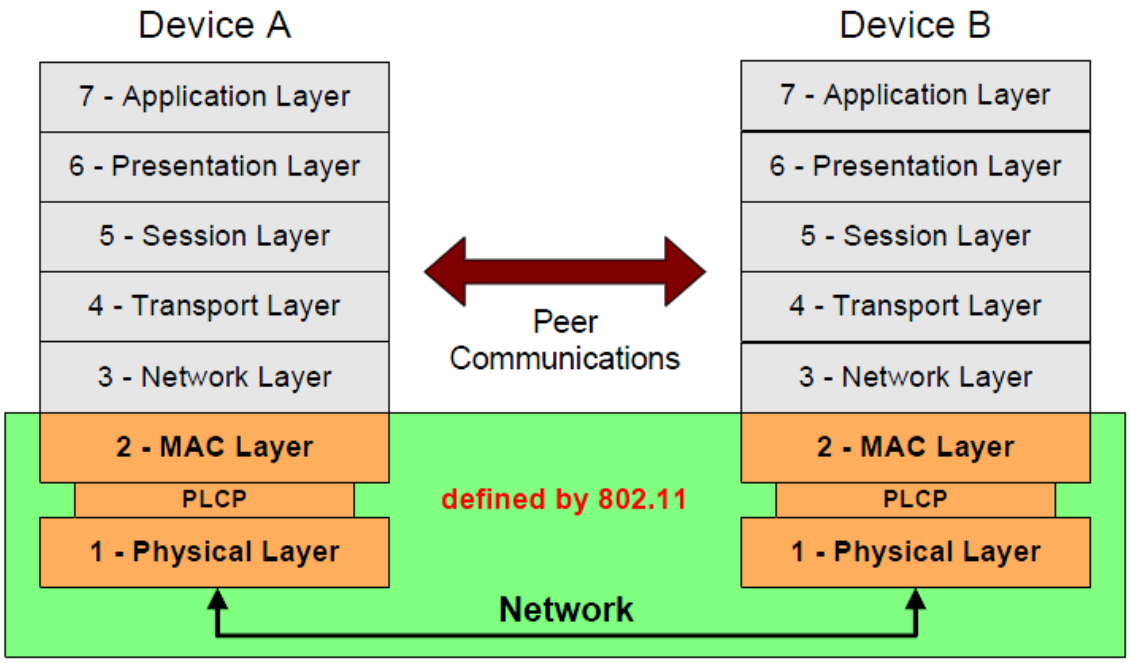

fig. 26 – Modelo OSI aplicando 802.11 (retirado de [7])

No caso do 802.11p, que é uma emenda do 802.11 de 2007, tem como base na sua camada física a modulação OFDM para a banda dos 5.9GHz já presente nas emendas a e g do 802.11 para a banda dos 5GHz. A *PHY* do 802.11p é essencialmente igual à que se pode encontrar na secção 17 da já conhecida norma IEEE 802.11a [22]. Na emenda p apenas são propostas pequenas alterações quer na *PHY* quer na *MAC* para garantir conexões robustas e rápidas configurações para veículos em movimento.

3.2.1 Arquitectura de referência do 802.11p

A camada física pode ser dividida em duas sub camadas, a PHY PLCP e PHY PMD, sendo a camada PLCP responsável por receber as primitivas provenientes da MAC, *TXVECTOR* e *RXVECTOR,* respectivamente para transmissão e recepção de tramas OFDM, pela sub camada PMD. Podemos dizer também que a PLCP é uma camada intermédia entre a MAC, e a PHY, que simplifica os processos de interacção entre estas.

3.2.1.1 PHY PLCP

Esta camada, como já foi dito, faz a ligação entre a MAC e a sub camada PHY PMD. A função desta camada passa essencialmente por receber as primitivas provenientes da MAC (TXVECTOR, RXVECTOR), e com base nos campos provenientes nestes vectores peencher correctamente a trama para ser transmitida pela PMD, como se pode ver esquematizado na figura 27.

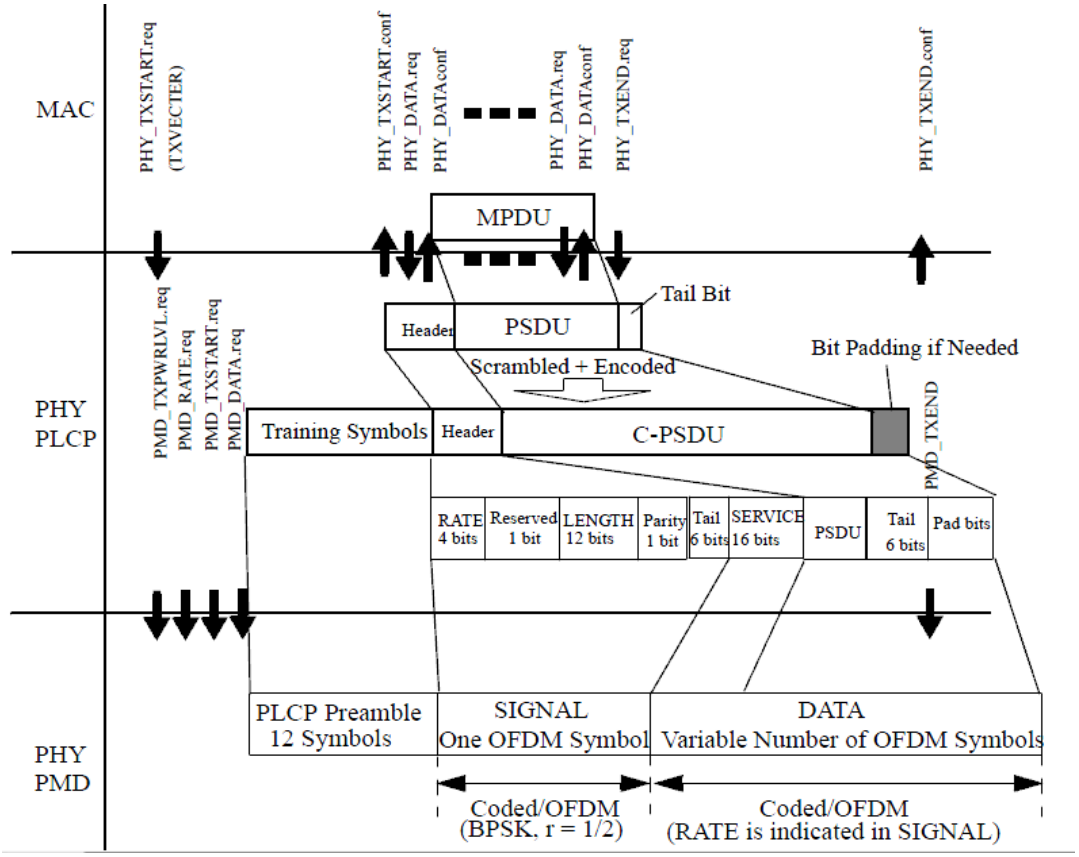

fig. 27 – Transmissão PLCP (retirado de [22])

3.2.1.2 PHY PMD

A sub camada PMD é então a camada responsável pela transmissão e recepção para/do meio das tramas OFDM. Esta é então constituída pelo FEC Coder (*Forward Error Correction Coder*), que introduz um código convolucional aos bits da trama, de seguida temos o *interleaving* dos bits. De seguida entramos na parte correspondente à modulação OFDM, onde se mapeiam os dados nas constelações definidas, de seguida faz-se o processamento da IFFT, com a introdução do prefixo cíclico nos símbolos OFDM. Por fim temos a parte de RF, onde os sinais IQ são transformados num sinal adequado à transmissão para o meio, é feito o *up conversion*, para os 5.9 GHz de transmissão e a amplificação para a potência de transmissão.

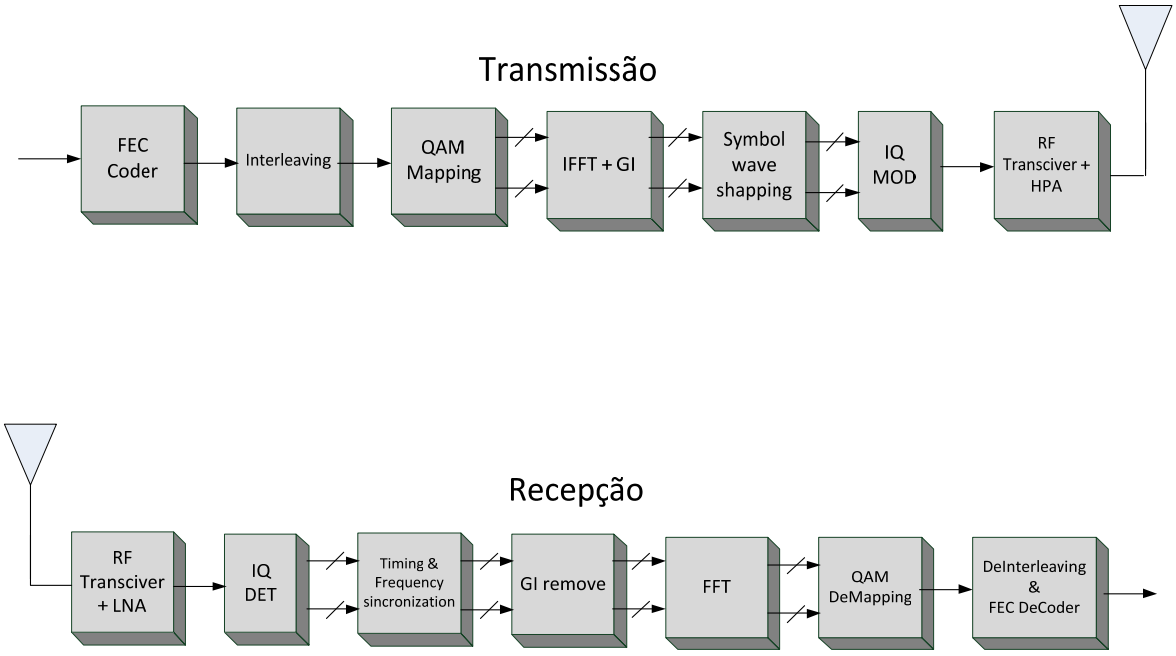

fig. 28 – Esquema de blocos da camada física 802.11p

Na recepção temos obviamente as operações inversas. Começando na recepção do meio dos sinais, estes são amplificados pelo LNA (*Low Noise Amplifier*), são sincronizados no tempo e na frequência, e a partir daqui as operações inversas realizadas na transmissão são executadas, começando pela remoção do prefixo cíclico, seguido pela FFT, o desmapeamento, o *deinterleaving* e o *fec decoder*, de modo a ter trama recebida pronta para passar para a MAC.

No 802.11p estão definidas quatro classes de potência para transmissão, que resultam em quatro máscaras espectrais que tem de ser cumpridas na transmissão, representadas na figura 29 e na tabela 1.

Tabela 1 – Classes de potência para canal de 10 MHz

Classe de potência de transmissão	Densidade espectral de potência permitida, dBr				
	± 4.5 MHz (± f1)	± 5.0 MHz (± f2)	± 5.5 MHz (± f3)	± 10 MHz (± f4)	± 15 MHz (± f5)
Classe A	0	-10	-20	-28	-40
Classe B	0	-16	-20	-28	-40
Classe C	0	-26	-32	-40	-50
Classe D	0	-35	-45	-55	-65

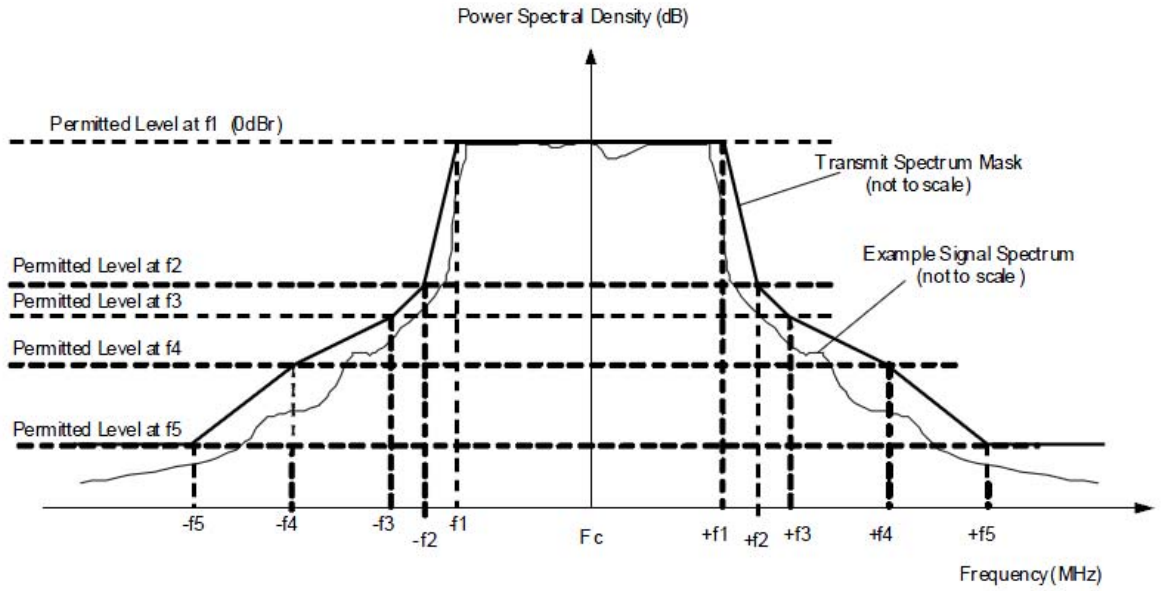

fig. 29 – Máscara espectral IEEE 802.11p (retirado de [23])

3.2.2 Funcionalidades

No 802.11 são definidos três modos de funcionamento, são eles o modo 20 MHz, 10 MHz e 5Mhz, podendo os diferentes modos ser conseguidos alterando a frequência de amostragem. No 802.11a é usual trabalhar-se com o modo de 20 MHz, e neste modo é capaz de providenciar velocidades de 6, 9, 12, 18, 24, 36, 48 e 54 Mb/s. Pode também usar o modo "*half-clocked*", ou em português a meio clock, que usa os já referidos 10 MHz de largura do canal, ou ainda é

possível "*quarter-clocked*", a um quarto do clock, para os 5 MHZ, sendo as velocidades transmissão possíveis nestes casos, respectivamente, metade e um quarto das referidas para 20MHz. Se no 802.11a é usual usar-se o modo de 20 MHz, no 802.11p o usual é usar-se o "*half-clocked*", de modo a ter sinais mais robustos. As principais diferenças entre 802.11a e p são apresentadas na tabela 2

Tabela 2 – Comparação entre IEEE 802.11a e IEEE 802.11p

Parâmetros	IEEE 802.11a	IEEE 802.11p	Mudança
Bit rate (Mb/s)	**6, 9,12,18, 24, 36, 48, 54 (obrigatórias – 6, 12, 24)**	**3, 4.5, 6, 9, 12, 18, 24, 27 (obrigatórias – 3, 6, 12)**	**Metade**
Tipo de Modulação	**BPSK, QPSK, 16QAM, 64QAM**	**BPSK, QPSK, 16QAM, 64QAM**	**Nada**
Code rate	**1/2, 2/3, 3/4**	**1/2, 2/3, 3/4**	**Nada**
Nº de portadoras	**52**	**52**	**Nada**
Espaçamento de frequência das portadoras	**0.3125 MHz**	**0.15625 MHz**	**Metade**
Duração do símbolo OFDM (T_{FFT}+T_{GI})	**4 µs**	**8 µs**	**Dobro**
T_{FFT}	**3.2 µs**	**6.4µs**	**Dobro**
T_{GI}	**0.8 µs**	**1.6 µs**	**Dobro**
Duração do preambulo	**16 µs**	**32 µs**	**Dobro**
Frequência de operação	**5GHz**	**5.9GHz**	

O sistema OFDM que constitui a camada é constituído por 48 sub portadoras para dados, sendo estas moduladas em BPSK, QPSK, 16QAM ou 64QAM, cujas constelações podem ser vistas na figura 34 do capítulo 4. A estas portadoras, juntasse outras quatro portadoras piloto, sendo estas moduladas em BPSK, e que servem como um padrão de dados utilizado no receptor para detecção e correcção de desvios de fase e/ou frequência, ou seja este sistema OFDM, é constituído por 52 portadoras com dados mapeadas nas posições (-26..-1,1…26), ocupando as portadoras piloto as posições ±21 e ±7. Mas para uma mais fácil implementação, ou seja uma implementação recorrendo a uma IFFT, o sistema completo aplica 64 portadoras (-32 .. 31) sendo as restantes portadoras não utilizadas, e consequentemente iguais a zero. A polaridade das portadoras em cada símbolo, é controlada pelo vector P[0..126]. A contribuição das portadoras no símbolo *n* é dada pela transformada inversa de fourier da sequência $P_{-26,\ 26}$, multiplicada por P[n]. Se n for maior que 127 (o tamanho do vector P), este é estendido ciclicamente até ao valor necessário. Um exemplo das polaridades aplicadas pode ser visto na tabela 3.

$P_{-26,\,26}$ = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, 0, 0}

P[0..126] = {1,1,1,1, -1,-1,-1,1, -1,-1,-1,-1, 1,1,-1,1, -1,-1,1,1, -1,1,1,-1, 1,1,1,1, 1,1,-1,1,

1,1,-1,1, 1,-1,-1,1, 1,1,-1,1, -1,-1,-1,1, -1,1,-1,-1, 1,-1,-1,1, 1,1,1,1, -1,-1,1,1,

-1,-1,1,-1, 1,-1,1,1, -1,-1,-1,1, 1,-1,-1,-1, -1,1,-1,-1, 1,-1,1,1, 1,1,-1,1, -1,1,-1,1,

-1,-1,-1,-1, -1,1,-1,1, 1,-1,1,-1, 1,1,1,-1, -1,1,-1,-1, -1,1,1,1, -1,-1,-1,-1, -1,-1,-1}

Tabela 3 – Polaridade das portadoras nos diferentes símbolos OFDM

n	Símbolo OFDM	Piloto em -21	Piloto em -7	Piloto em 7	Piloto em 21	P[n]
0	Signal	1.0+0j	1.0+0j	1.0+0j	-1.0+0j	1
1	Data 1	1.0+0j	1.0+0j	1.0+0j	-1.0+0j	1
2	Data 2	1.0+0j	1.0+0j	1.0+0j	-1.0+0j	1
3	Data 3	1.0+0j	1.0+0j	1.0+0j	-1.0+0j	1
4	Data 4	-1.0+0j	-1.0+0j	-1.0+0j	1.0+0j	-1
5	Data 5	-1.0+0j	-1.0+0j	-1.0+0j	1.0+0j	-1
6	Data 6	-1.0+0j	-1.0+0j	-1.0+0j	1.0+0j	-1

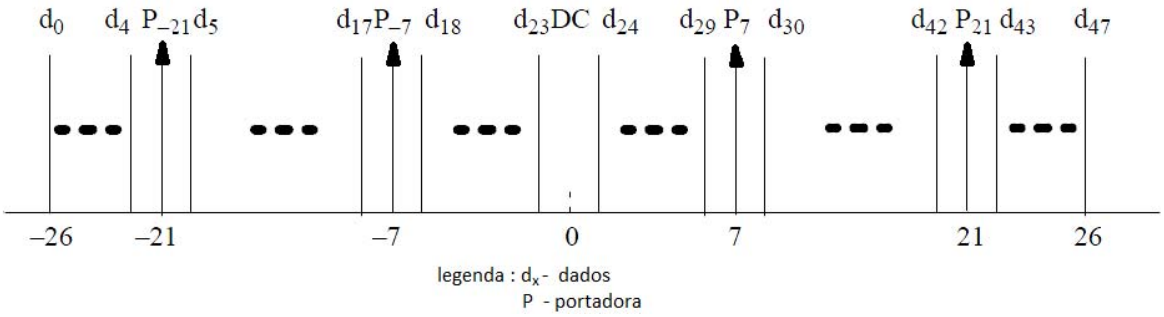

fig. 30 – Distribuição das portadoras piloto (retirado de [22])

Para atingir as várias taxas de transmissão, referidas anteriormente, modulações diferentes com códigos redundantes com diferentes taxas de codificação são introduzidos de acordo com a tabela 4.

Tabela 4 – Parâmetros dependentes na modulação.

Modulação	Taxa de codificação	Nº de bits por portadora	Nº bits por símbolo OFDM	Bits de dados por símbolo OFDM	Data rate para canal de 10 MHZ (Mb/s)
BPSK	1/2	1	48	24	3
BPSK	3/4	1	48	36	4.5
QPSK	1/2	2	96	48	6
QPSK	3/4	2	96	72	9
16QAM	1/2	4	192	96	12
16QAM	3/4	4	192	144	18
64QAM	2/3	6	288	192	24
64QAM	3/4	6	288	216	27

3.2.3 Formato da trama 802.11p

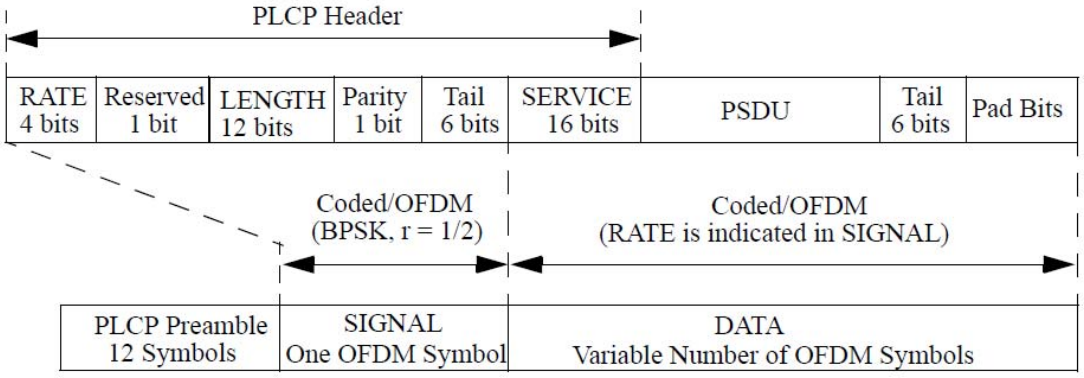

Fig. 31 – Trama PLCP (retirado de [22])

O formato da trama a ser transmitida está exemplificada na figura 31, onde podemos ver que existem três campos distintos, e são eles o *PLCP Preamble*, que é constituído pelas chamadas sequências de treino, que se dividem pelas sequências curtas, e pelas sequências longas, ocupando no total da transmissão 12 símbolos OFDM. Estas sequências servem para ajustar os parâmetros da camada *PHY* no receptor, para que este possa receber a trama correctamente. Ajustes como por exemplo, aquisição temporal, ver desvios na frequência para corrigi-los,

estimação do canal entre outros, são conseguidos fazendo a análise do *PLCP Preamble* no receptor.

De seguida na trama temos o campo chamado de SIGNAL, campo este bastante importante quer para a PHY transmissora, quer para a PHY receptora. Neste campo estão definidos os parâmetros que os blocos que constituem a PHY precisam de conhecer para o processamento correcto da trama. Por exemplo os 4 bits do campo sub campo RATE, que indicam o ritmo de transmissão. O ritmo de transmissão define o tipo de modulação, e a taxa de codificação do código convolucional a usar, como se pode ver na tabela 5. O campo SIGNAL é transmitido num símbolo OFDM, e devido à sua importância para uma correcta recepção, este é sempre modulado em BPSK, com código redundante de 1/2, que corresponde à mínima taxa de transmissão que a norma permite, mas que tem a vantagem de ser mais facilmente bem recebido e descodificado no receptor como se pretende, mesmo nas condições de transmissão mais adversas.

No terceiro e último campo temos então os dados que a MAC quer enviar (MPDU), já devidamente colocado pela PLCP no formato de transmissão. Este campo tem como seria de esperar um número variável de símbolos OFDM, pois depende do tamanho dos dados que se pretende enviar, que está definido no campo SIGNAL no sub campo LENGTH.

Tabela 5 – Valores dependentes do campo RATE

Campo RATE (bits)	Taxa de transmissão(Mb/s) (canal de 10MHz)	Modulação	Taxa de codificação
1101	3	BPSK	1/2
1111	4.5	BPSK	3/4
0101	6	QPSK	1/2
0111	9	QPSK	3/4
1001	12	16QAM	1/2
1011	18	16QAM	3/4
0001	24	64QAM	2/3
0011	27	64QAM	3/4

4 Transmissão OFDM em 802.11p

4.1 Introdução

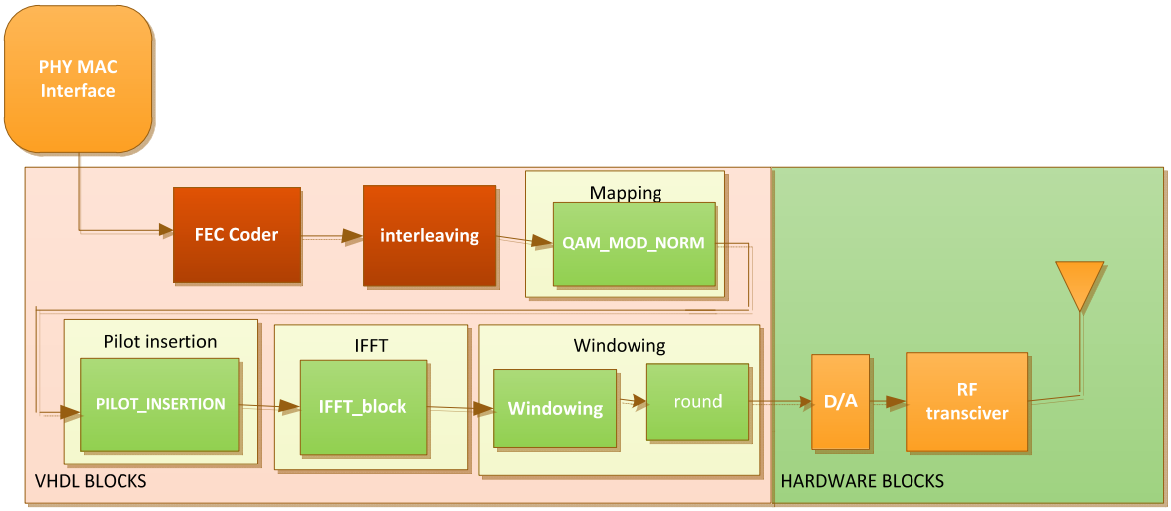

fig. 32 – Blocos da transmissão

Como foi visto no terceiro capítulo a camada física do 802.11p tem a configuração que pode ser vista em esquema de blocos na figura 32. No esquema de blocos estão representados os principais blocos constituintes da transmissão e onde está especificado quais os que são implementados na FPGA recorrendo ao código VHDL. No mesmo esquema temos também diferenciada a parte dos blocos VHDL que foram alvo da minha implementação (*Mapping*, *Pilot Insertion*, *IFFT*, e *Windowing*), e que vão ser descritos neste capítulo.

4.1.1 Abordagem de implementação

Para implementar estes blocos, a abordagem que vou seguir é a de começando no bloco de *mapping*, fazer toda a sua implementação e validação, e após validado comportamentalmente avançar para o seguinte bloco da cadeia, e assim sucessivamente. Para a validação comportamental, recorro a *test benchs* adequadas, e verifico o funcionamento com a ajuda da ferramenta ISIM do ISE. No final da cadeia de transmissão é esperado ver uma trama completa

modulada de acordo com a norma. Para comparar resultados, no anexo G da norma 802.11a, que como já foi dito anteriormente, é compatível com a p. No referido anexo tem um exemplo completo com vários passos de transmissão de uma trama. No final da cadeia de transmissão esta deverá estar de acordo com o exemplo da norma [24].

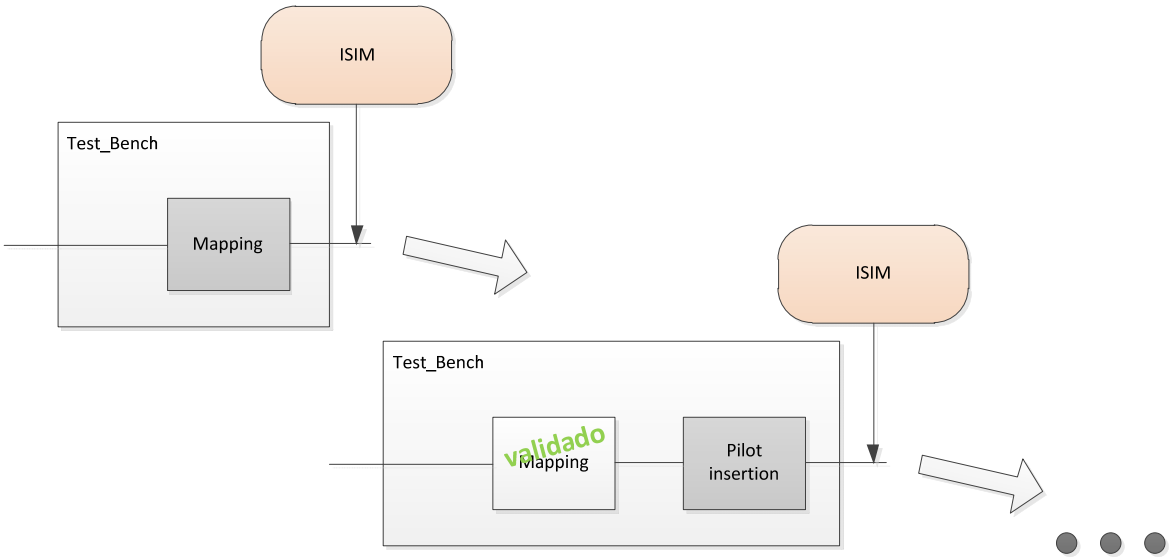

fig. 33 – Abordagem de implementação e validação

4.2 Mapeamento e Normalização

Como podemos ver no esquema de blocos da figura 32, o primeiro bloco que aparece a verde, é o bloco de *mapping*, ou em português mapeamento. Neste bloco é feito o mapeamento da *stream* de bits que sai do bloco de *interleaving*. Este mapeamento pode ser feito numa das quatro modulações seguintes, que dependem do ritmo de transmissão escolhido, e que está definido no campo RATE, da trama PLCP da figura 31:

- BPSK
- QPSK
- 16-QAM
- 64-QAM

Que têm as seguintes constelações em código *gray*, e onde o bit b_0 é o primeiro que aparece na bit *stream*:

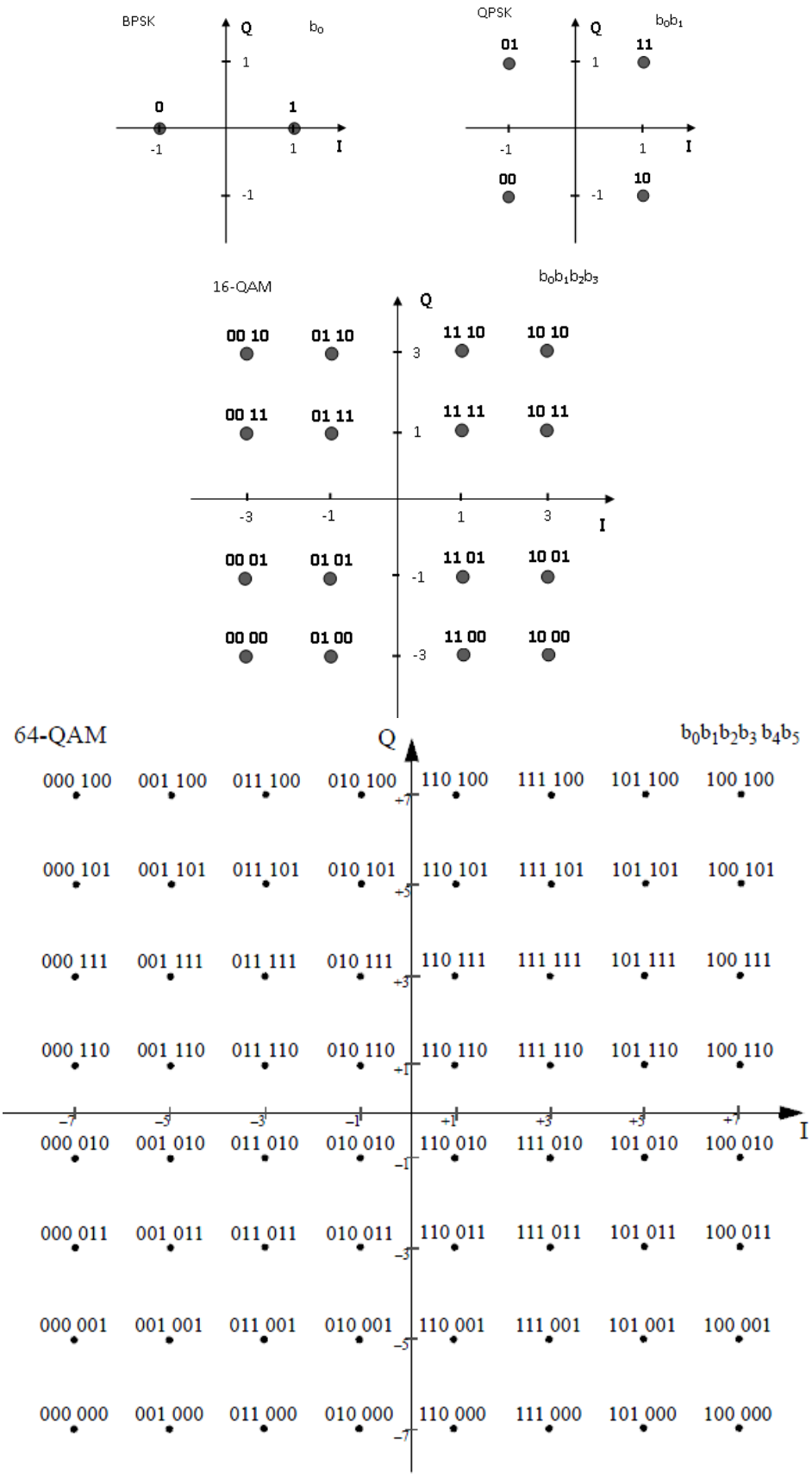

fig. 34 – Constelações BPSK, QPSK, 16QAM, 64QAM definidas pela norma 802.11p

Das constelações podemos retirar as seguintes tabelas de correspondência dos padrões de bits entrada com o valor de fase e quadratura a atribuir a esse padrão:

Tabela 6 – Tabela de mapeamento BPSK

Bits de entrada b_0	I	Q
0	-1	0
1	1	0

Tabela 7 – Tabela de mapeamento QPSK

Bits de entrada b_0	I
0	-1
1	1

Bits de entrada b_1	Q
0	-1
1	1

Tabela 8 – Tabela de mapeamento 16QAM

Bits de entrada b_0b_1	I
00	-3
01	-1
11	1
10	3

Bits de entrada b_2b_3	Q
00	-3
01	-1
11	1
10	3

Tabela 9 – Tabela de mapeamento 64QAM

Bits de entrada $b_0b_1b_2$	I
000	-7
001	-5
011	-3
010	-1
110	1
111	3
101	5
100	7

Bits de entrada $b_3b_4b_5$	Q
000	-7
001	-5
011	-3
010	-1
110	1
111	3
101	5
100	7

E portanto para fazer este mapeamento, foi criado o bloco em VHDL com o nome "QAM_modulator", apresentado em detalhe no ponto 4.2.1.

Depois de mapeados os dados I, Q resultantes sofrem uma normalização de maneira a formar símbolos do tipo:

$$d = (I + jQ) \times K_{MOD}$$

Em que o parâmetro K_{MOD} é dependente da modulação, e está definido na tabela 11. A normalização serve para que todas as constelações tenham a mesma potência média. O valor de K_{MOD} pode na prática ser ligeiramente diferente, desde que sejam respeitados pelo dispositivo os valores de precisão de modulação que estão representados na tabela 10 [22].

Tabela 10 – Erro relativo da constelação permitido pela norma

Erro relativo (dB)	Modulação	Taxa de codificação
-5	BPSK	1/2
-8	BPSK	3/4
-10	QPSK	1/2
-13	QPSK	3/4
-16	16QAM	1/2
-19	16QAM	3/4
-22	64QAM	2/3
-25	64QAM	3/4

Tabela 11 – Parâmetro dependente K_{MOD}

Modulação	K_{MOD}
BPSK	1
QPSK	$1/\sqrt{2}$
16QAM	$1/\sqrt{10}$
64QAM	$1/\sqrt{42}$

Para fazer a operação de normalização foi criado o bloco em VHDL com o nome “normalization”, apresentado detalhadamente no ponto 4.2.2.

4.2.1 Bloco QAM_modulator

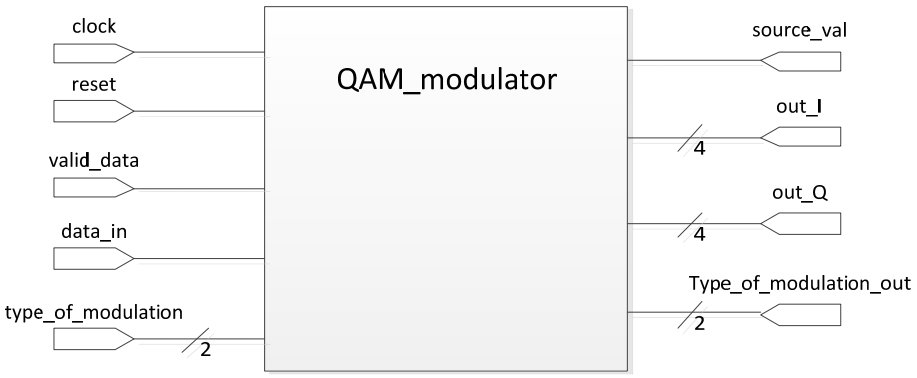

fig. 35 – Bloco QAM_modulator

Tabela 12 – Interfaces do bloco QAM_modulator

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
valid_data	Entrada	Sinal que indica que os dados que provêm do bloco anterior são válidos
data_in	Entrada	*Bit stream* de dados a serem mapeados
type_of_modulation	Entrada 2 bits	Sinal que indica o tipo de modulação pretendida
source_val	saida	Sinal que indica ao bloco seguinte que os dados são válidos
out_I	Saída de 4 bits	Saída de dados correspondentes à fase
out_Q	Saída de 4 bits	Saída de dados correspondentes à quadratura
Type_of_modulation_out	Saída de 2 bits	Tipo de modulação dos dados na saída

Apresentadas as portas do bloco, este funciona da seguinte forma:

- Quando o sinal de dados válidos é activo alto significa que os dados que estão presentes na entrada data_in, são válidos e tem de ser modulados, de acordo com a norma.

- Dependendo do tipo de modulação a *stream* de dados é dividida em N_{bpsc}, (número de bits por portadora), que será 1, 2, 4 ou 6 caso se trate da modulação BPSK, QPSK, 16QAM ou 64QAM respectivamente.

- O sinal type_of_modulation de dois bits, que correspondem aos dois bits menos significativos do sub-campo RATE, do campo SIGNAL da trama PLCP, indicam qual o tipo de modulação a ser usada nos dados, e consequentemente o valor de N_{bpsc}:

- 11 → BPSK → $N_{bpsc} = 1$
- 01 → QPSK → $N_{bpsc} = 2$
- 10 → 16QAM → $N_{bpsc} = 4$
- 00 → 64QAM → $N_{bpsc} = 6$

- Depois de correctamente agrupados os bits, as saídas out_I, e out_Q, são colocadas nos valores devidos, por exemplo, se tivermos type_of_modulation = 00, e a sequência de bits de entrada for '011001', temos recorrendo às tabelas de mapeamento que foram apresentadas no ponto 5.2:

  011 → out_I = -3 ('1101' em binário complemento para 2)
  001 → out_Q = -5 ('1011' em binário complemento para 2)

- Quando são atribuídos os valores correctos a out_I, e out_Q, o sinal source_val fica activo alto para indicar ao bloco seguinte que os dados estão válidos.

Este bloco tem ainda em conta, que o tipo de modulação pode ser diferente ao longo de uma trama, visto que a norma diz que o conteúdo correspondente ao campo SIGNAL, que vai corresponder a um símbolo OFDM, com as portadoras moduladas em BPSK, independentemente do ritmo de transmissão que é indicado para transmissão. Tendo em conta isto este bloco tem um contador interno, que indica se está no campo SIGNAL, e a modulação deve ser BPSK e não o que está indicado no sinal type_of_modulation. Por este motivo este bloco tem a saída type_of_modulation_out, que quando a trama está no campo SIGNAL, é posto no valor '11', e quando está fora, toma o valor de type_of_modulation. Isto evita que no bloco seguinte se tenha de verificar novamente se está ou não campo SIGNAL.

4.2.1.1 Validação comportamental do QAM_modulator

Para a validação comportamental do bloco, foi realizada uma test_bench, onde uma trama perfeitamente usada apenas para efeitos de dos testes, trata-se de um padrão de 48 bits que se repete pelo tempo da simulação, e faz o papel de data_in, o período do relógio da simulação é de

10 ns. As entradas são então definidas de maneira a verificar o funcionamento do bloco. O resultado está apresentado nas figuras seguintes:

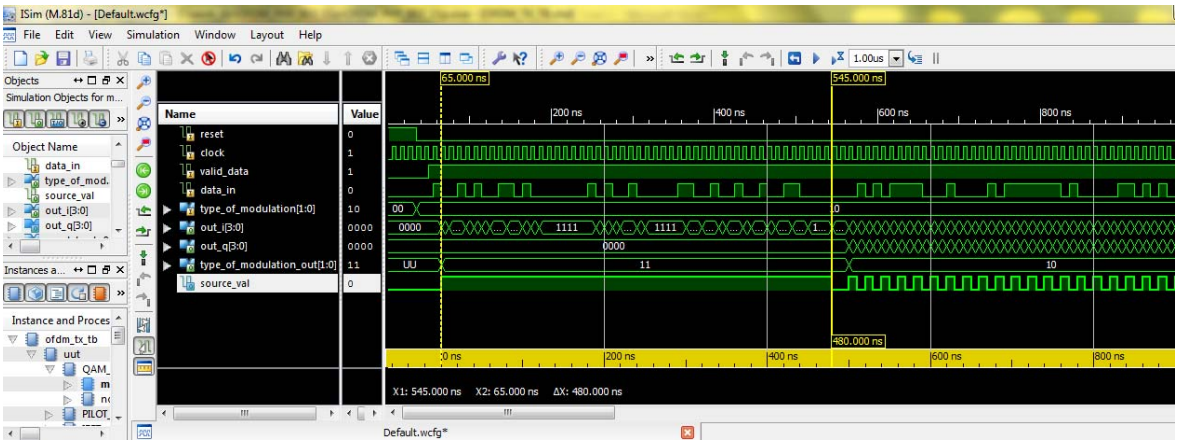

fig. 36 – Vista geral do comportamento do bloco QAM_modulator

Na figura 36 pretendo mostrar uma vista alargada, do funcionamento do bloco, de maneira a dar especial atenção ao sinal source_val que se apresenta seleccionado na figura, e que mostra claramente o mapeamento correspondente ao SIGNAL, ou seja os 480 ns (48 bits) em que o sinal se encontra ininterruptamente activo alto, e de seguida, podemos ver o comportamento do sinal na parte dos dados, sendo que nesta simulação a modulação que está a ser feita é QPSK, e como tal os dados na saída só são validos após terem sido "vistos" dois bits da *stream* de entrada, o que não acontece em BPSK onde 1 bit na entrada corresponde a um símbolo na saída do bloco.

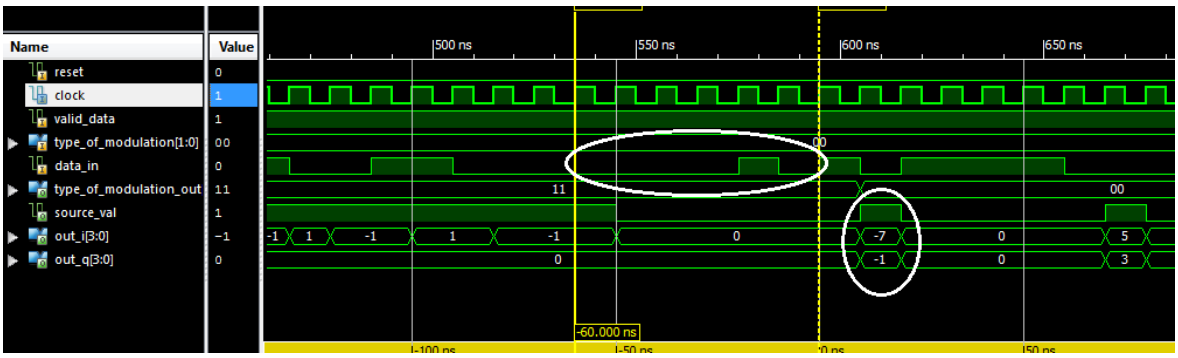

fig. 37 – Exemplo de com modulação 64QAM

Na figura 37, podemos ver o mapeamento feito, neste caso em 64QAM, onde rodeados com o balão branco maior se encontram realçados os seis bits da entrada data_in que foram mapeados, e no outro balão branco estão realçadas as saídas out_I e out_Q com o valor do mapeamento,

bem como o sinal de source_val que indica ao bloco seguinte que estes são dados válidos. Podemos ainda notar que este zoom foi feito numa zona de transição entre os bits do campo SIGNAL, e os bits do campo DATA. Pode ver-se então nesta figura a mudança no sinal de type_of_modulation_out, que retrata essa transição.

Após uma verificação exaustiva de toda a trama, até que esta se voltasse a repetir, e consequentemente as saídas se voltassem a repetir, verifiquei bit a bit o funcionamento correcto do bloco.

4.2.2 Bloco Normalization

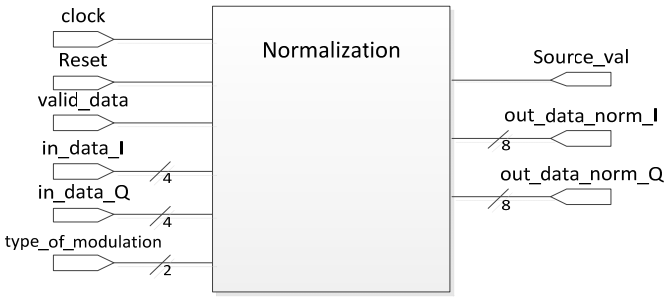

fig. 38 – Bloco Normalization

Tabela 13 – Interfaces do bloco Normalization

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
valid_data	Entrada	Sinal que indica que os dados que provêm do bloco anterior são válidos
type_of_modulation	Entrada 2 bits	Sinal que indica o tipo de modulação em que se encontram os dados nas entradas in_data_I e in_data_Q
In_data_I	Entrada 4 bits	Bits correspondentes à fase
In_data_Q	Entrada 4 bits	Bits correspondentes à quadratura
source_val	Saída	Sinal que indica ao bloco seguinte que os dados são válidos
out_data_norm_I	Saída de8 bits	Saída de dados normalizados correspondentes à fase
out_data_norm_Q	Saída de 8 bits	Saída de dados normalizados correspondentes à quadratura

A função deste bloco, como já foi dito, é a de normalizar os valores das constelações para que estas tenham uma potência média igual. Este bloco tem de fazer basicamente a multiplicação das entradas in_data_I e in_data_Q, pelo factor dependente da modulação K_{MOD}, cujos valores estão apresentados na tabela 11 do capítulo 4. Mas visto que o parâmetro K_{MOD} é fixo para cada modulação, e atendendo a que os valores do mapeamento, os valores que I e Q podem tomar são apenas -7, -5, -3, -1, 1, 3, 5 e 7, ou seja apenas podem tomar oito valores distintos, a multiplicação destes pelo factor de normalização foi feita fora do contexto da programação VHDL, tendo sido passados para esse contexto os oito resultados das multiplicações, convertidos para 8 bits em binário complemento para dois. Assim para executar a tarefa a que se destina, apenas precisa de olhar para as entradas in_data_I, in_data_Q e type_of_modulation, e atribuir às saídas out_data_norm_I e out_data_norm_Q os valores correspondentes previamente calculados, e deste modo poupar recursos da FPGA, nomeadamente dois multiplicadores.

Deste modo se compararmos a funcionalidade deste bloco com a do bloco anterior, podemos verificar que estas são perfeitamente semelhantes, e como tal estar a usar estes dois blocos é estar a usar mais recursos que os necessários, e assim estes dois blocos foram fundidos e deram origem ao bloco com o nome QAM_MOD_NORM, que se apresenta no ponto 4.2.3.

4.2.2.1 Validação comportamental do bloco "Normalization"

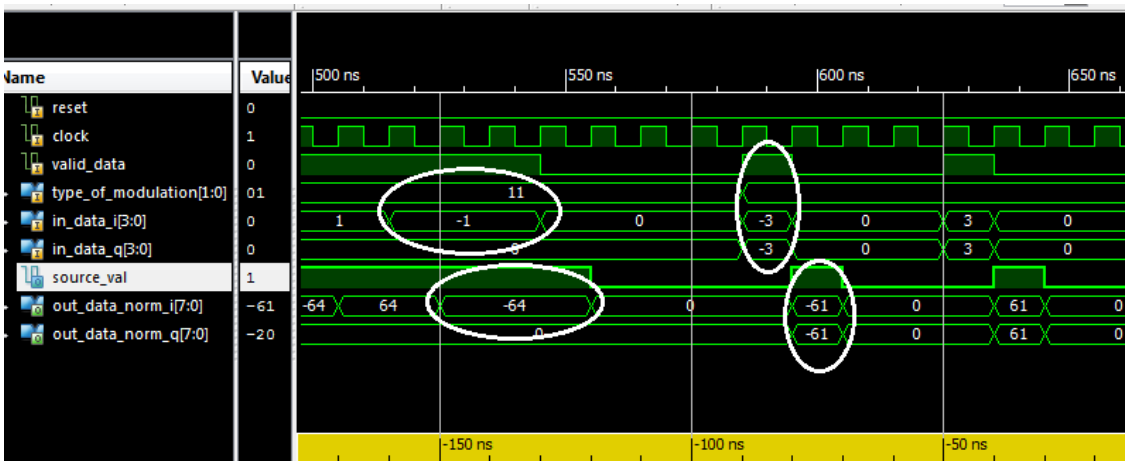

fig. 39 – Exemplificação das entradas e correspondentes saídas do bloco normalization

Na figura 39, podemos ver dois exemplos do funcionamento do bloco. Estamos numa zona em que se pode ver a transição entre o campo SIGNAL, e o campo DATA, que está nesta simulação modulado em 16QAM. Podemos ver que com um ciclo de relógio de atraso, as saídas são actualizadas com o valor normalizado das entradas. Para uma melhor visualização estes foram convertidos para decimal.

4.2.3 Bloco QAM_MOD_NORM

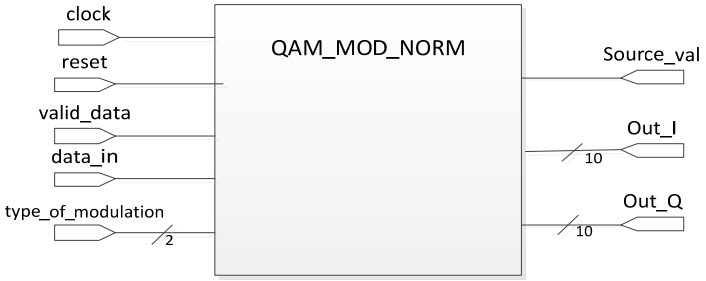

fig. 40 – Bloco QAM_MOD_NORM

Tabela 14 – Interfaces do bloco QAM_MODULATOR

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
valid_data	Entrada	Sinal que indica que os dados que provêm do bloco anterior são válidos
data_in	Entrada	*Bit stream* de dados a serem mapeados
type_of_modulation	Entrada 2 bits	Sinal que indica o tipo de modulação pretendida
source_val	saida	Sinal que indica ao bloco seguinte que os dados são válidos
out_I	Saída de 10 bits	Saída de dados normalizados correspondentes à fase
out_Q	Saída de 10 bits	Saída de dados normalizados correspondentes à quadratura

Este bloco é a fusão dos blocos QAM_modulator e o Normalization, e a sua maneira de operar, é a mesma que foi descrita anteriormente para o bloco QAM_modulator, mas neste bloco depois de comparados os grupos de bits de cada modulação, são atribuídos às saídas out_I e out_Q os valores de mapeamento já normalizados.

4.2.3.1 Validação comportamental do bloco "QAM_MOD_NORM"

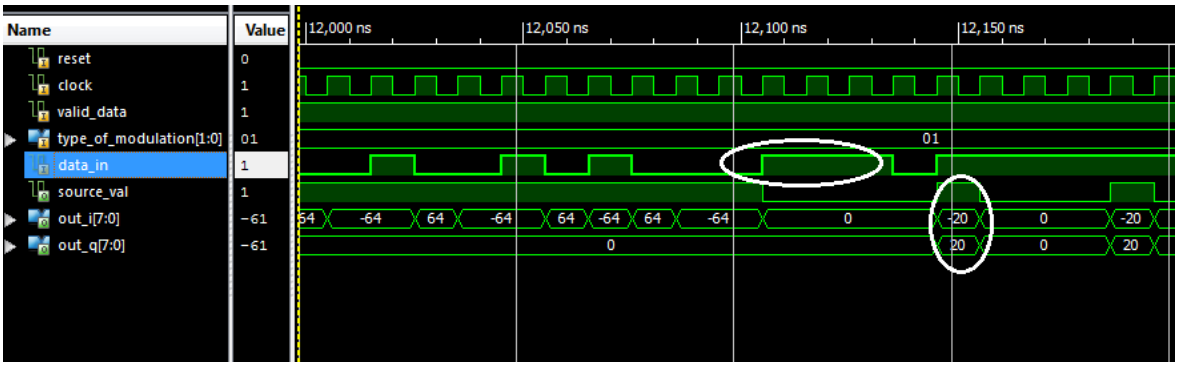

fig. 41 – Exemplo de codificação 16QAM com o bloco QAM_MOD_NORM

Como referido anteriormente pode ser verificado na figura acima o funcionamento do bloco, que funde o funcionamento dos dois blocos anteriores, à *stream* de entrada do sinal data_in, é feito mapeamento com a normalização correcta já aplicada.

4.3 Inserção das Frequências Piloto

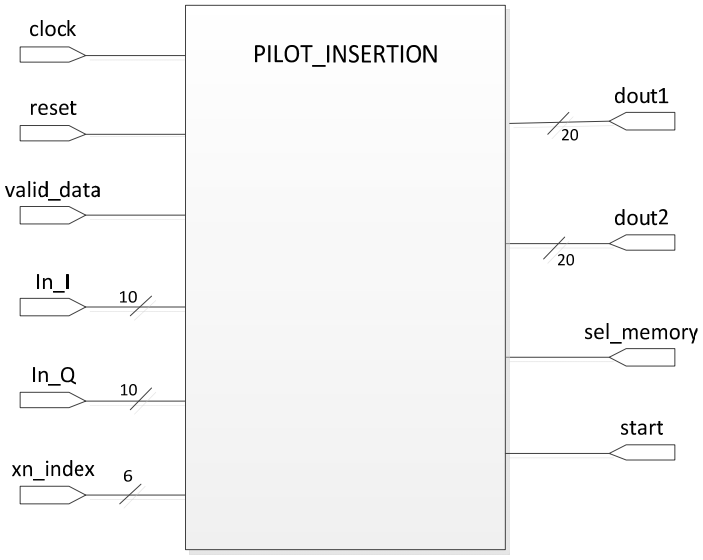

fig. 42 – Bloco PILOT_INSERTION

Tabela 15 – Interfaces do bloco PILOT_INSERTION

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
valid_data	Entrada	Sinal que indica que os dados que vem do bloco anterior são válidos
in_I, in_Q	Entradas de 10 bits	Entradas de dados de dez bits (in_I – fase, in_Q - quadratura)
xn_index	Entrada de 6 bits	Sinal de endereçamento para a leitura das memórias de saída, que provêm do bloco IFFT
dout1, dout2	Saídas de 20 bits	Saídas de dados de 20 bits dos memórias de saída, que contêm os dados referentes à fase e quadratura
start	Saída	Sinal que indica ao bloco seguinte que pode começar a leitura da memória de saída.
sel_memory	Saída	Sinal usado pelo bloco seguinte para que seja escolhida a memória correcta para a leitura

Este bloco tem como principal função a inserção das frequências piloto nas posições indicadas na figura 30 do capítulo 3. E para além desta função faz ainda o mapeamento das portadoras para as posições correctas a serem usadas pelo bloco de IFFT. Para a realizar esta função este bloco é formado por um fifo de entrada e duas memórias de 64 posições na saída, realizados com recurso a IPcores da XILINX, onde se podem definir as suas características recorrendo é ferramenta *core generator* existente no ISE. Para além destes *IP cores*, temos a parte que faz realmente a inserção e o mapeamento nas memórias das frequências piloto. O esquema de blocos pode ser visto na figura seguinte.

Visto que este bloco se situa antes da IFFT, e esse bloco é feito a partir de um IPcore da XILINX, e tem um funcionamento que tem de ser cumprido de maneira a conseguir o correcto cálculo da IFFT, este bloco para além da inserção das frequências piloto, cria também a interface necessária para o uso correcto do bloco seguinte. Uma das funções extra que este faz, é inserir as frequências não usadas, que como foi dito no ponto 3.2.2, a IFFT tem um comprimento N de 64, mas apenas 52 das portadoras são usadas, sendo quatro delas as portadoras piloto. Devido a esta situação e à inserção das piloto, temos que por cada 48 símbolos de dados que chegam ao bloco, deste saem 64 (figura 44). Isto para o caso de estarmos na modulação BPSK, em que 48 ciclos de relógio chegam 48 símbolos. Devido a esta situação foi colocado um primeiro fifo, a que se deu o nome de fifo_in, na entrada que armazena os dados de entrada, para dar tempo ao Pilot_insertor de fazer a sua tarefa. Este fifo na pior situação de crescimento, ou seja na modulação em BPSK, cresce 16 posições por cada 48 símbolos BPSK, o que equivale a 48 bits. A norma define que o tamanho máximo dos pacotes é de 4095 bytes, ou seja 32760 bits. Fazendo as contas de crescimento no pior dos casos de modulação (16 por cada 48), isto obriga um tamanho do fifo de 20x10920! Este valor é enorme e consumiria todos os recursos de memória da FPGA que está a ser usada. Na prática os pacotes geralmente não ultrapassam os 1500 bytes, o que já daria apenas 4000 posições para o fifo, o que continua a ser grande. Assim para os efeitos de demonstração pretendidos as tramas vão ser de tamanho reduzido, sendo a trama de teste presente no anexo da norma de 1200 bits, e pela qual foram feitos a maioria dos testes.

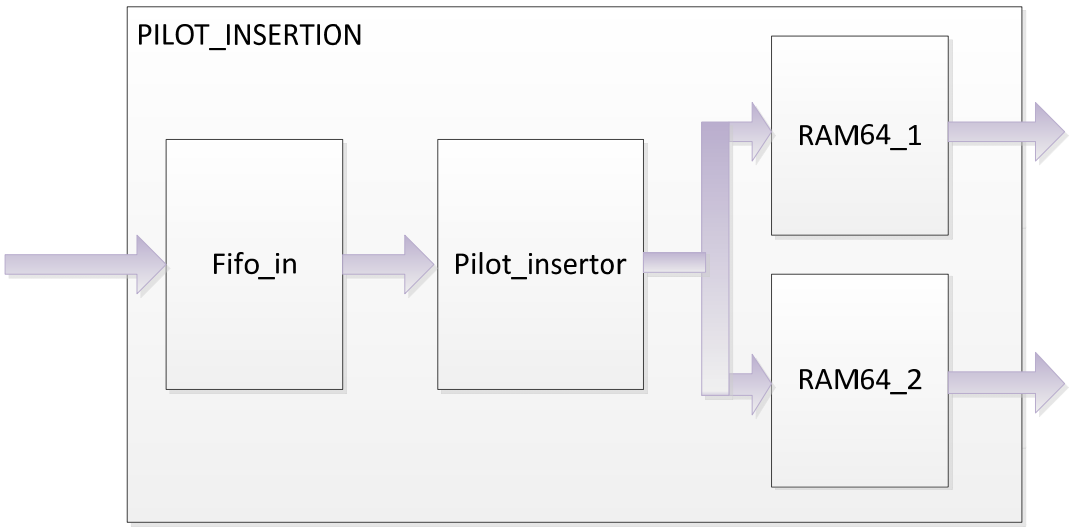

fig. 43 – Esquema de blocos do bloco top model PILOT_INSERTION.

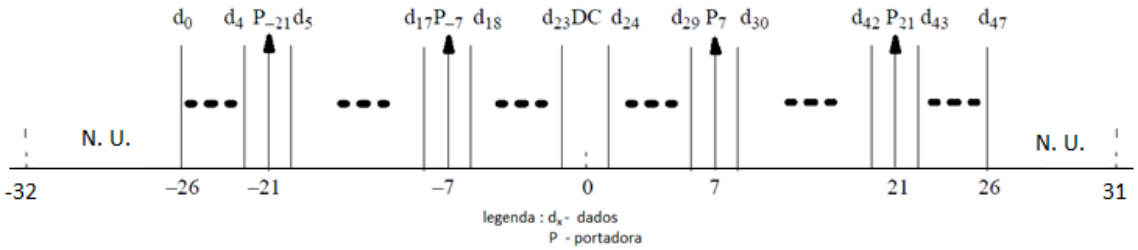

fig. 44 – Distribuição de frequências no símbolo OFDM à saída do bloco PILOT_INSERTOR

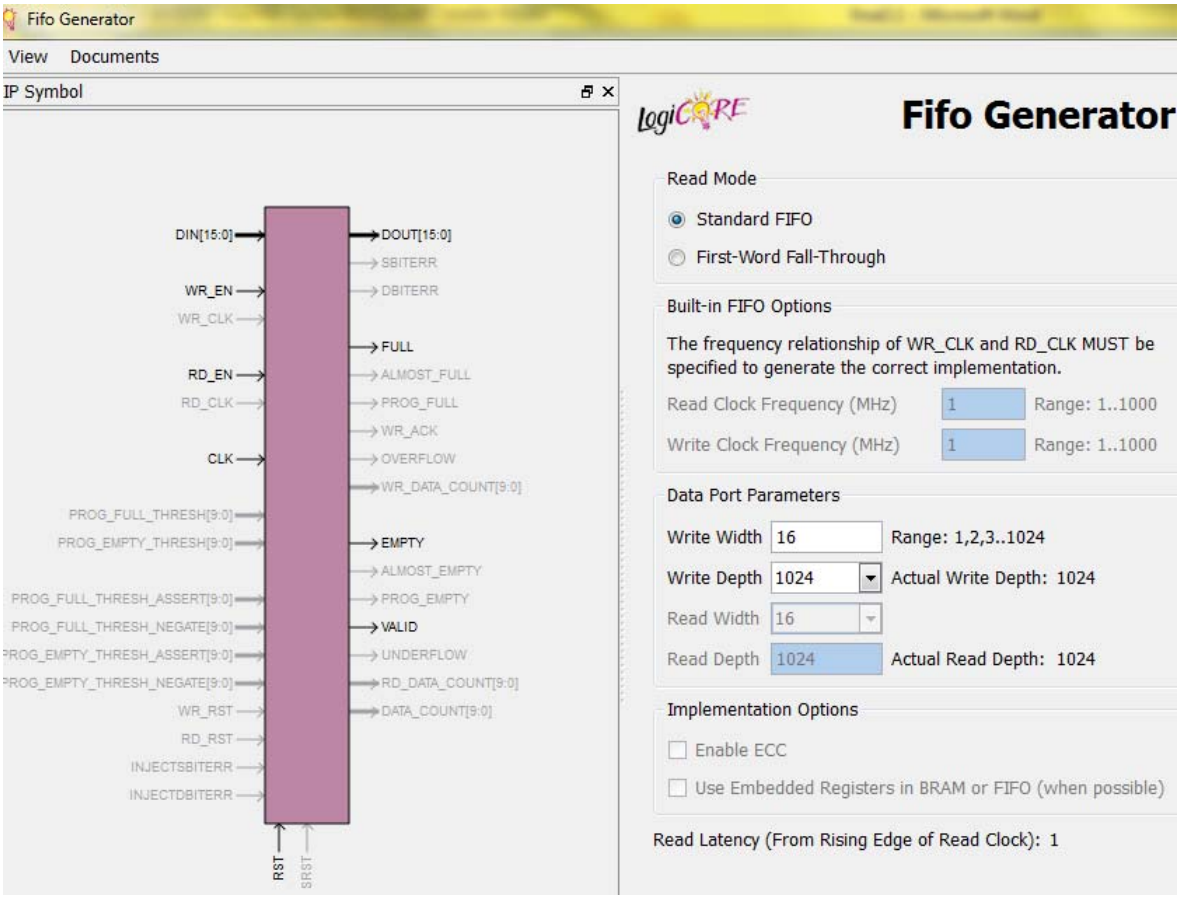

fig. 45 – Esquema do IPcore do fifo_in

. O fifo_in tem as seguintes características, escolhidas no *core generator*:

- O tamanho do fifo é de 20 x 1024.
- Uma entrada assíncrona de reset, coincidente com o reset do sistema
- Um sinal de relógio igual ao do relógio do sistema
- Uma entrada/saída de dados de 20 Bits, onde se ligam as entradas in_I e in_Q do bloco PILOT_INSERTION, sendo que in_I corresponde aos bits 19 a 10, e in_Q aos bits 9 a 0 do barramento de dados do fifo (DIN/DOUT).
- Tem ainda *flags* de *full*, de *empty*, e de valid sendo que esta indica quando é que os dados na saída após um pedido de leitura são válidos.
- Uma entrada para a leitura de dados (rd_en), que é activa pelo sub bloco pilot_insertor, quando este está pronto para receber novos dados.
- Uma entrada para escrita (wr_en), que liga ao sinal de valid_data do bloco *top model* onde se insere este fifo.

Portanto o bloco PILOT_INSERTION funciona numa primeira fase do seguinte modo:

- Os dados válidos provenientes do bloco anterior (QAM_MOD_NORM) são armazenados no fifo_in.

- O sub bloco pilot_insertor começa por colocar nas saídas I e Q, as portadoras de dados correspondentes a -26 a -22 como se pode ver na figura 44. Estas primeiras portadoras, são pedidas ao fifo_in. Quando chegar à posição -21, em que insere a primeira portadora piloto, não são pedidos novos dados ao fifo_in, voltando a ser pedidos novos dados imediatamente a seguir até chegar a posição da próxima piloto. O processo repete-se de modo a ter nas saídas as portadoras como exemplifica a figura 44.

O bloco de IFFT, requer que os dados após o início dos cálculos, a cada ciclo de relógio, requer novos dados válidos nas suas entradas durante todos os N valores do seu comprimento, sem qualquer interrupção. Embora o bloco de IFFT tenha a opção de uma entrada de *clock enable* (CE) que poderia ser activada quando na saída do bloco pilot_insertion, não houvesse dados válidos. No entanto esta solução foi testada e não se mostrou de grande utilidade visto que com *clock enable* aplicado faz com que todo o bloco congele, e portanto se estiverem a sair dados nesse momento estes também são parados, o que compromete o funcionamento do restante circuito para a frente. Isto poderia ser corrigido alterando a cadeia para a frente, mas o uso do CE fica mesmo impossibilitado devido à inserção do prefixo cíclico no bloco de IFFT, que faz com que o core de IFFT fique durante algum tempo desabilitado de receber novos dados. Devido a isto e ao funcionamento das diferentes modulações, em que neste caso temos o pior caso na modulação 64QAM, em que a cada portadora correspondem seis ciclos de relógio, isto faz com que os dados na saída do sub bloco pilot_insertor, não sejam contínuos. Por este facto foram introduzidas duas memórias iguais de 64 posições, para guardar todas as portadoras de um símbolo OFDM. São necessárias duas pois quando uma delas fica completamente preenchida, o bloco pilot_insertor dá sinal ao bloco seguinte (IFFT) para começar a fazer as operações e a despejar os dados, o que demora algum tempo. Durante esse tempo os novos dados que chegam entretanto são armazenados na outra memória. E o processo repete-se durante toda a trama. Assim os parâmetros escolhidos para as memórias são:

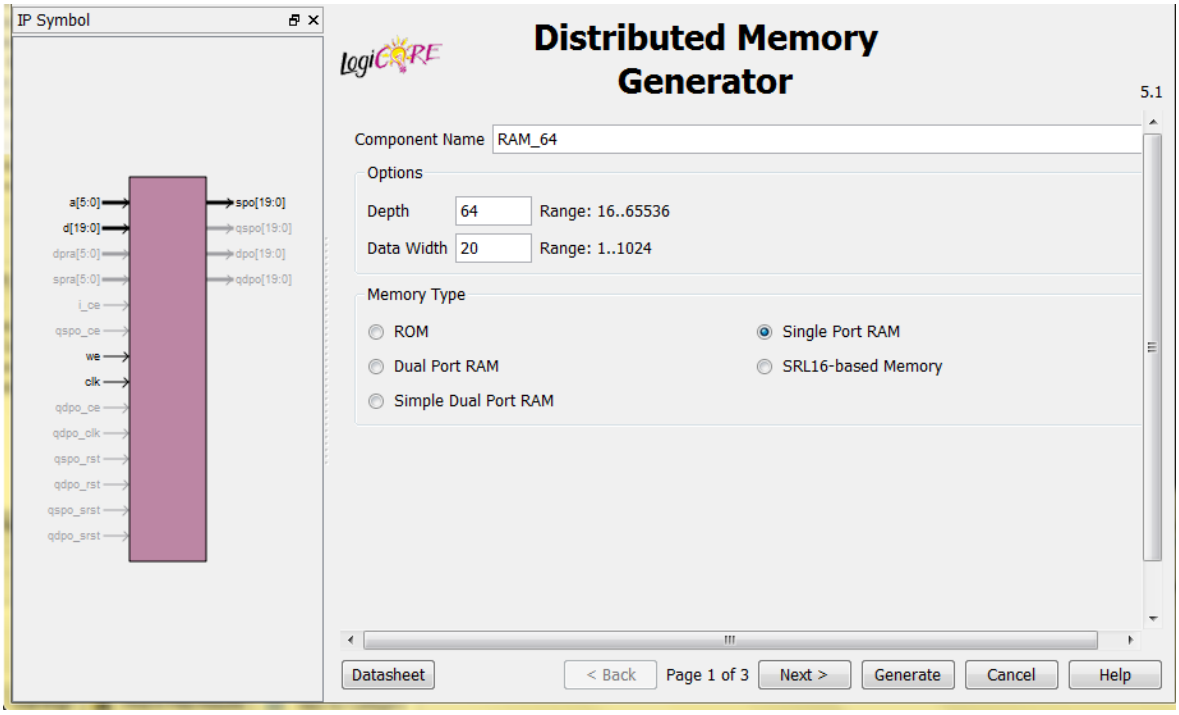

fig. 46 – Esquema do IPcore para o fifo_out1 e fifo_out2

- O tamanho das memórias RAM é 20 x 64.
- Um sinal de relógio coincidente com o do sistema.
- Uma entrada/saída de dados de 20 Bits, onde se ligam as saídas I e Q do bloco pilot_isertor, sendo que I corresponde aos bits 19 a 10, e Q aos bits 9 a 0 do barramento de dados das memórias (d/spo).
- O barramento de dados, liga-se fisicamente ao sinal xn_index, que é uma saída do bloco seguinte, e que fica encarregue de endereçar a leitura das memórias.

No que diz respeito à parte de escrita nas memórias de saída, para que se possa utilizar o sinal xn_index, proveniente da IFFT, para leitura os dados têm que ser mapeados nas memórias de modo a cumprir a especificação de cálculo da IFFT. A norma indica que para o cálculo da IFFT, as portadoras -26 a -1 da figura 44, deverão ser mapeados nas entradas 38 a 63 do core IFFT, sendo as portadoras 0 a 26 mapeadas nas mesmas entradas do core. As restantes entradas não utilizadas (27 a 37) do core, contemplam as portadoras não utilizadas da figura 44. O esquema de mapeamento (escrita nas memórias de saída), pode ser visto na tabela seguinte. Desta forma o sinal xn_index proveniente da IFFT e que indica qual a entrada a ser lida pelo core, pode ser usada para a leitura e endereçamento da memória, sendo cumpridas as especificações da norma.

Tabela 16 – Esquema de mapeamento das memórias

RAM_64	
ADDRESS	CONTENT
0	#NULL
1	#1
2	#2
.	.
.	.
26	#26
27	#NULL
.	.
37	#NULL
38	#-26
.	.
.	.
62	#-2
63	#-1

O sub bloco pilot_insertor, está então encarregado de fazer este mapeamento. O bloco *top model* PILOT_INSERTION, fica encarregado de fazer a ligação e o controlo de escrita na memória correcta, entre o bloco pilot_insertor e as memórias. A leitura das memórias, é controlada pelo bloco seguinte, estando este encarregue de escolher a memória correcta de onde tem de ler os dados, isto recorrendo ao sinal de controlo sel_memory que este bloco lhe fornece.

4.3.1 Validação comportamental do bloco "PILOT_INSERTION"

Para a verificar o comportamento deste bloco com a *test bench*, este tem de estar conectado com o bloco IFFT_block, visto que como já foi dito é neste bloco que as operações de leitura das memórias de saída são endereçadas. Na prática podemos considerar o conjunto PILOT_INSERTOIN e IFFT_Block como um conjunto *master-slave*, em que o IFFT_block é o *master*. Tendo isto em consideração para a realização da test bench, apresentam-se de seguida os resultados.

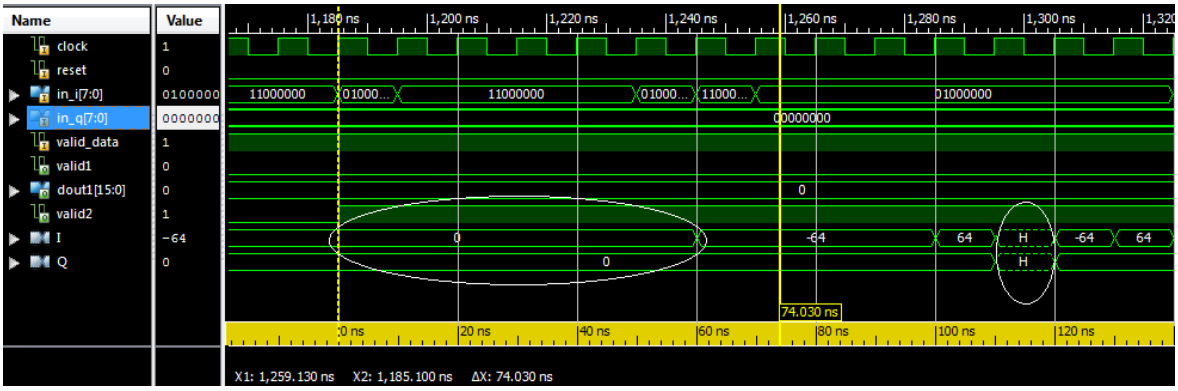

fig. 47 - Exemplo do funcionamento do bloco PILOT_INSERTION(1)

Para uma mais fácil visualização das piloto inseridas, colocou-se um valor de “H”, para mais facilmente poderem ser identificadas na simulação. Depois de verificado o correcto funcionamento foram colocados os valores correctos, ou seja a modulação BPSK. Na figura 48 podemos ver o início do símbolo correspondente ao SIGNAL, e onde se pretende dar relevo nos primeiros sinais rodeados das seis frequências não utilizadas nas posições -32 a -26. De seguida temos dados desde a posição -26 até -21 onde encontramos a primeira frequência piloto. Avançando na simulação foram verificadas todas as correctas posições das frequências piloto, bem como as polaridades das mesmas ao longo dos símbolos. O que mostra o correcto funcionamento e termos da função pretendida do bloco.

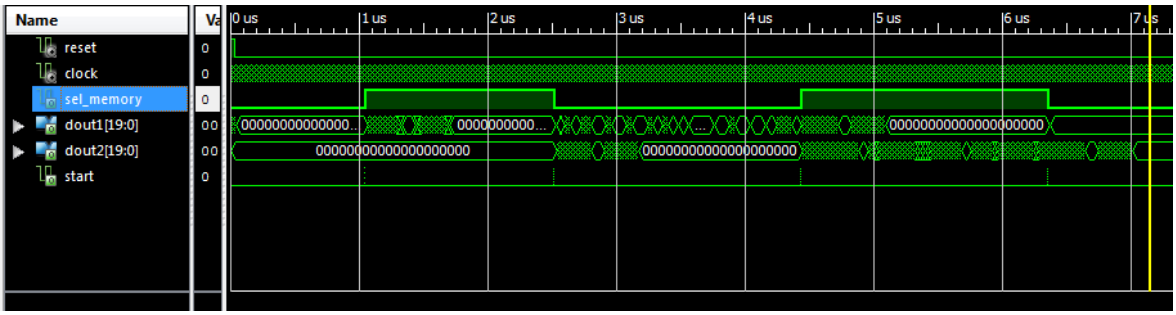

fig. 48 - Exemplo do funcionamento do bloco PILOT_INSERTION(2)

A figura 48, tem como objectivo mostrar o comportamento na saída do bloco. Pode ver-se na figura os símbolos a serem lidos alternadamente de cada uma das memórias de saída, através dos sinal sel_memory, que quando activo alto significa que os dados estão a ser lidos da memória um, quando activo baixo está a ser lida a memória dois.

4.4 IFFT e Prefixo Cíclico

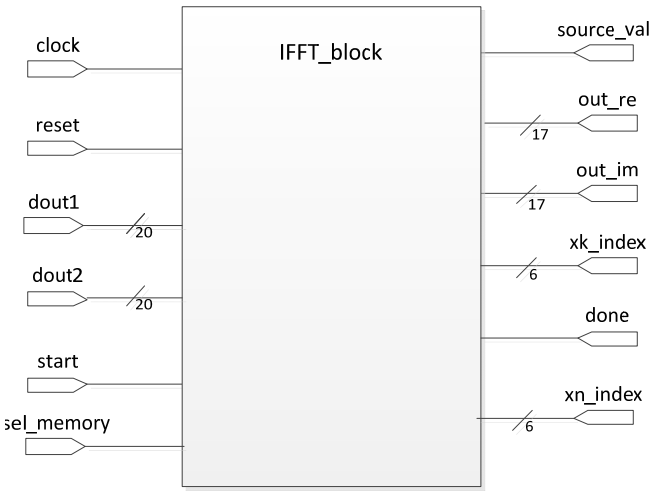

fig. 49 –Bloco IFFT_block

Tabela 17 - Interfaces do bloco IFFT_block

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
sel_memory	Entrada	Sinal que indica qual das entradas dout1 ou dout2 deve ser considerada
start	Entrada	Sinal de que indica que se devem começar a fazer o processamento da IFFT
dout1, dout2	Entradas de 20 bits	Dados referentes à fase e quadratura armazenados nas memórias (do bloco anterior, dai os nomes dout1 e dout2)
xn_index	Saída de 6 bits	Índice da amostra de entrada
done	Saída	Sinal que indica que os cálculos da IFFT foram feitos, e os dados vão começar a sair
xk_index	Saída de 6 bits	Índice da amostra de saída
source_val	Saída	Sinal que indica ao bloco seguinte que os dados são válidos
out_re	Saída de 17 bits	Saída de dados correspondente à parte real
out_im	Saída de 17 bits	Saída de dados correspondente à parte imaginária

Este é o bloco onde o cálculo da IFFT é feito de acordo com a expressão

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{jnk\pi/N} \qquad ,n = 0, \dots, N-1$$

Para o efeito este é constituído por lógica de controlo, para fazer a ponte entre bloco anterior, e a parte essencial deste bloco, que é o IPcore que realiza a IFFT. Este IPcore, fornecido pelo XILINX, tem o nome de *fast fourier transform 7.1*, e tem quatro modos distintos de operação. A escolha de um dos quatro, é um compromisso entre consumo de recursos da FPGA e velocidade, como se pode ver na figura 50. Os modos são:

- Pipelined, Streaming I/O
- Radix-4, Burst I/O
- Radix-2, Burst I/O
- Radix-2 lite, Burst I/O

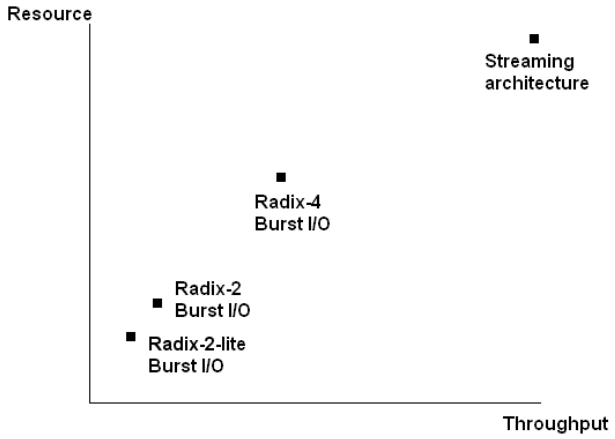

fig. 50 – Relação recursos vs velocidade, entre os modos de operação do IPcore fast fourier transform v7.1. (retirado de [25])

Como o cálculo da IFFT é uma parte crucial da cadeia de transmissão, o modo de operação escolhido foi o que consome mais recursos, mas que em contra partida é o que oferece melhor velocidade nos dados de saída, o modo *Pipelined, streaming* I/O. Para além da maior velocidade este método tem a grande vantagem de implementar automaticamente a extensão cíclica. E tem a vantagem de poder receber novos dados enquanto ainda está a calcular os anteriores, ou seja pode receber dados ininterruptamente, isto no caso onde a opção de extensão cíclica não seja

escolhida. No caso de extensão cíclica escolhida, um tempo de guarda equivalente ao tempo da extensão cíclica tem de ser dado, até poderem ser introduzidos novos valores nas entradas. Visto que neste modo os dados de entrada entram de forma série, não há necessidade com esta arquitectura de paralelizar os dados, e consequentemente após a IFFT, voltar a passá-los para dados série, visto que é na forma serie que entram e saem do bloco.

Assim os parâmetros escolhidos no *core generator* para a IFFT são:

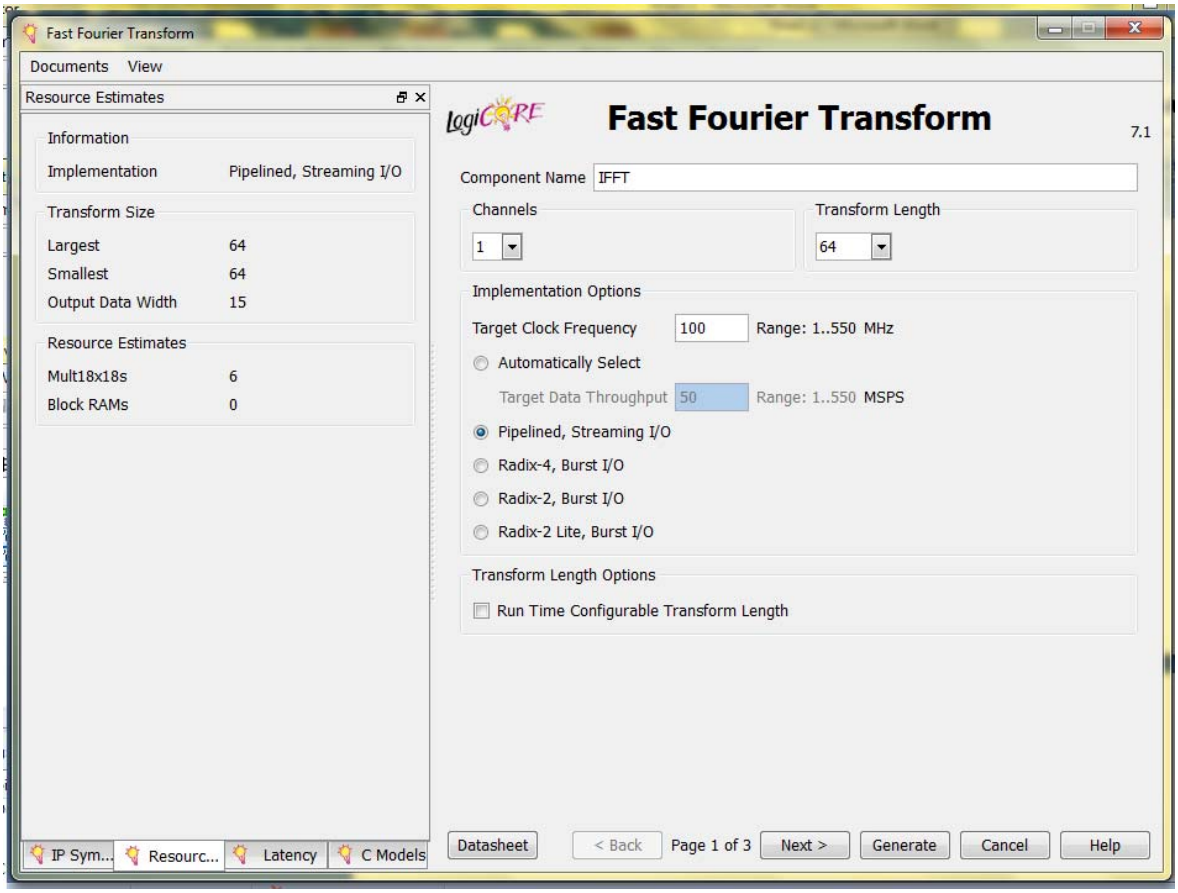

fig. 51 – Parâmetros escolhidos no Fast Fourier Transform core (1)

- Comprimento N = 64;
- *Pipelined, streaming* I/O;
- Foi escolhido um relógio de operação de 100MHz

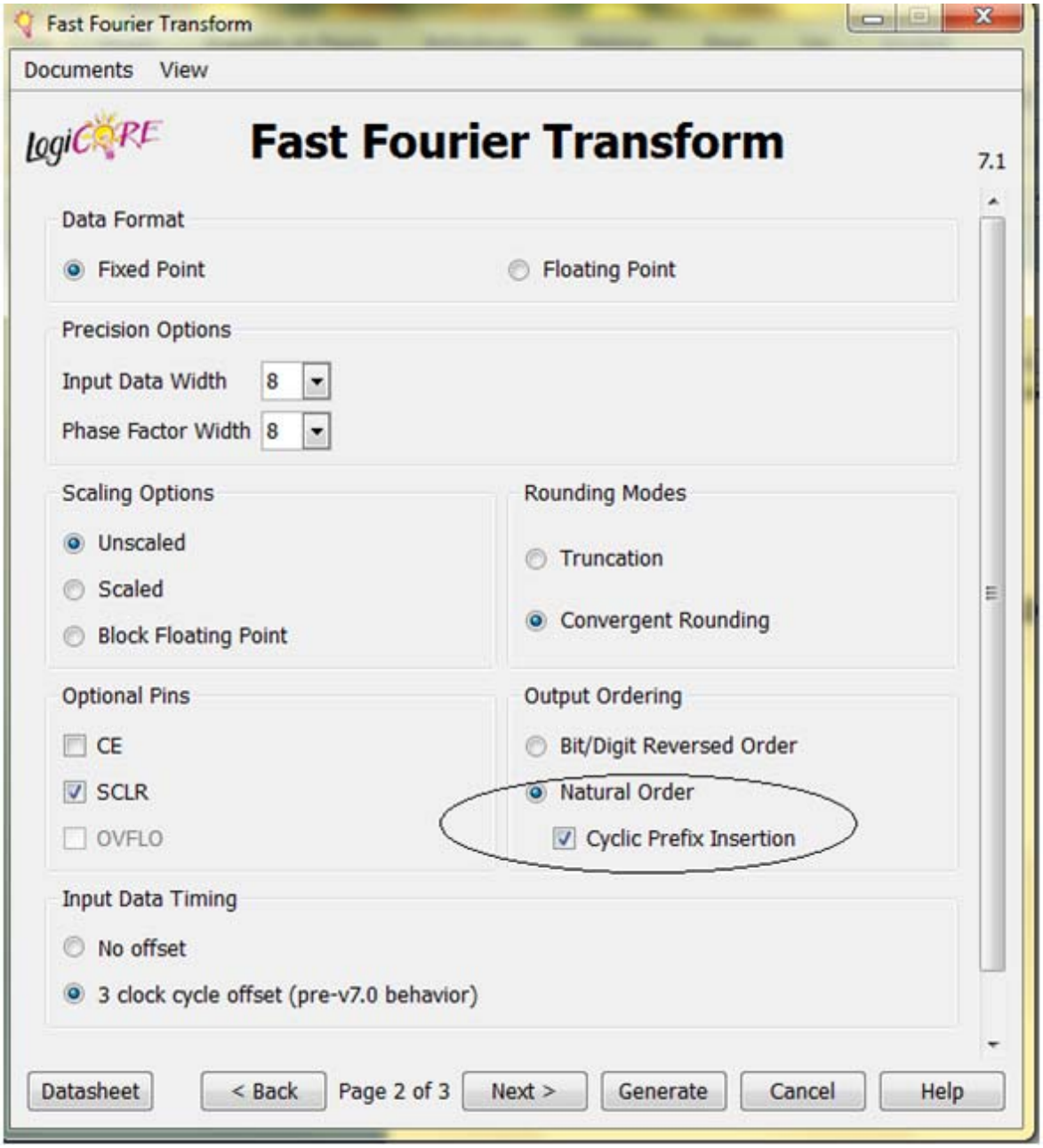

fig. 52 - Parâmetros escolhidos no Fast Fourier Transform core (2)

- Representação dos valores em *Fixed Point*
- Dados de entrada fase e quadratura de 10 bits
- Dados *Unscaled*, tem como consequência saídas real e imaginaria de 17 bits, mas evita *overflow* e consequentes erros.
- Arredondamento nos cálculos internos.
- *3 clock cycle offset* - Esta opção permite fazer a leitura de memórias através do sinal xn_index. O que faz essencialmente é atrasar 3 ciclos de relógio a amostra indicada pelo sinal xn_index, ou seja na prática os cálculos apenas se começam a fazer 3 ciclos de relógio após o sinal de start ser activado.

- Pino de reset assíncrono, correspondente ao reset do sistema
- No modo *pipelined* escolhendo a opção de os dados aparecerem na saída pela ordem natural, podemos escolher a opção de inserir o prefixo cíclico, como pode ser visto na figura 52

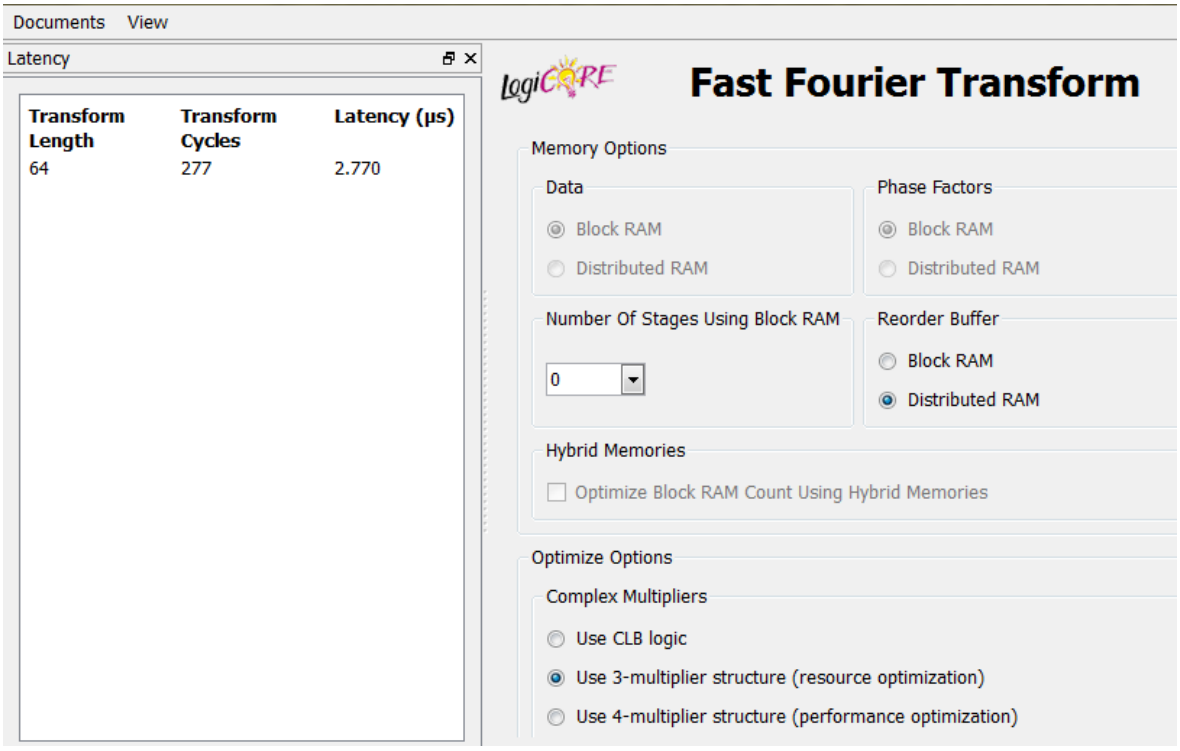

fig. 53 - Parâmetros escolhidos no *Fast Fourier Transform* core (3)

- Uso de estrutura de 3-multiplicadores, que é um meio-termo entre recursos velocidade de operação.

Com estas opções podemos ver na figura 53 que a latência no cálculo da IFFT é de 277 ciclos de relógio, ou seja desde que o primeiro valor entra até que seja colocado na saída o primeiro resultado, passam 277 ciclos. O componente gerado com estas opções tem então a seguinte configuração de entradas/saídas:

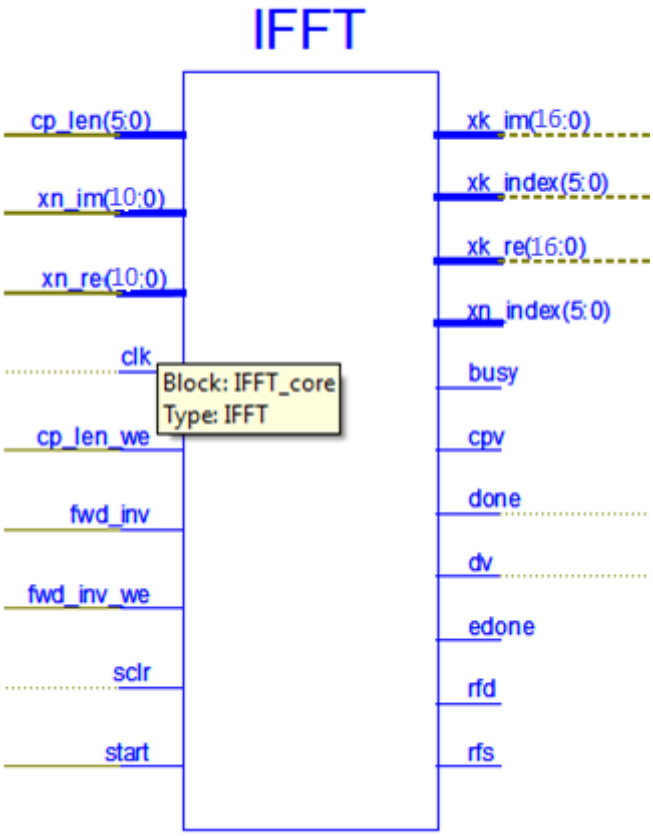

fig. 54 – Componente IFFT_core

O bloco IFFT_block é então formado, pelo IFFT_core, e pela parte de controlo deste. A parte de controlo tem como funções:

- Definir a operação a executar, visto que o IPcore, realiza a FFT ou a IFFT. Para executar IFFT, o sinal fwd_inv é utilizado. Quando definido a '1' esta realiza a FFT, quando '0' realiza a desejada IFFT deste ponto. O sinal fwd_inv_we é o sinal de *enable* do sinal fwd_inv.

- Escolher a memória de saída do bloco anterior (PILOT_INSERTION), da qual irá ser lido um conjunto completo de 64 portadoras, às quais vai ser feito o cálculo. Esta informação é dada pelo sinal de entrada sel_memory, que selecciona conforme seja '1' ou '0' o sinal dout correcto.

- Através do sinal xn_index endereçar as memórias de saída previamente mapeadas para a utilização deste sinal no bloco anterior.

- Dar início aos cálculos no core, através da activação do sinal start deste. Este sinal liga-se directamente ao sinal com o mesmo nome proveniente do PILOT_INSERTION.

- Fazer a divisão dos sinais dout1 e dout2 provenientes do PILOT_INSERTION. A divisão é simplesmente atribuir aos sinais correctos do IFFT_core, os dados que correspondem à fase, e à quadratura, a que correspondem os bits 19 a 10 e 9 a 0 dos sinais dout, respectivamente.

- Definir o tamanho do prefixo cíclico desejado, neste caso é 1/4 do comprimento da IFFT, ou seja 16. Esta informação é atribuída ao sinal cp_len do core. O sinal cp_len_we, é o sinal de *enable* do cp_len.

4.4.1 Validação comportamental dos blocos IFFT_block.

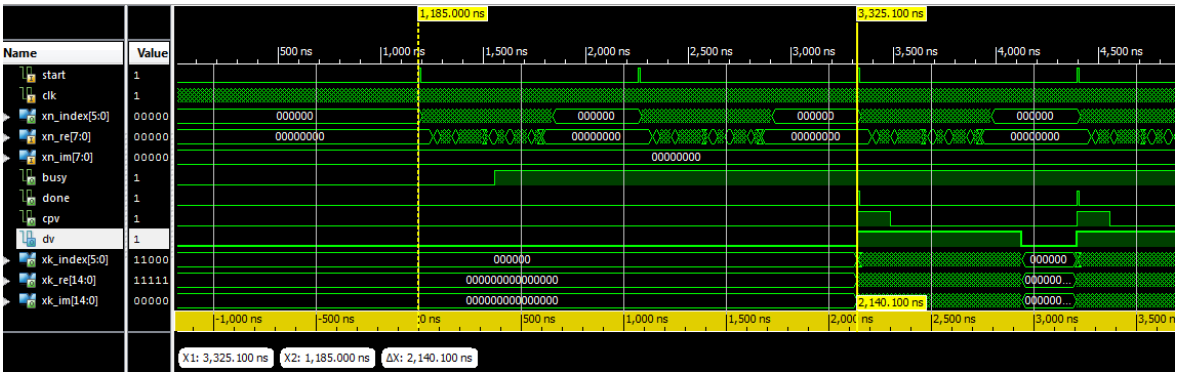

fig. 55 – Verificação do funcionamento do bloco IFFT_block(1)

Na figura 55 pode ser visto a latência do cálculo da IFFT (o tempo que passa desde que o primeiro valor entra até que seja actualizada a saída com o valor calculado), entre as duas linhas verticais, que é de 214 ciclos. Para todos os valores calculados temos mais 63 ciclos de relógio que vai aos 277 ciclos no total. Na figura pode ver-se o sinal de dados de saída validos, e consegue-se ver os diferentes símbolos OFDM, delimitados por este sinal. O sinal acima do seleccionado, é o sinal de prefixo cíclico válido, e pode ver-se a sua introdução no início do símbolo, como é desejado.

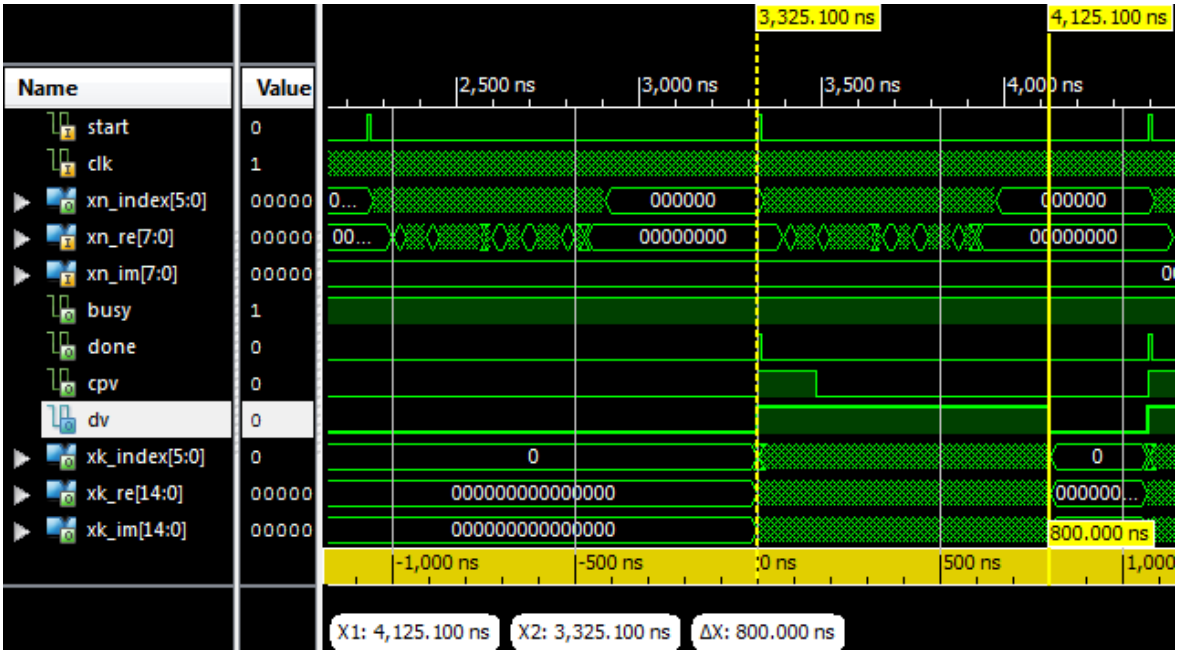

fig. 56 - Verificação do funcionamento do bloco IFFT_block(2)

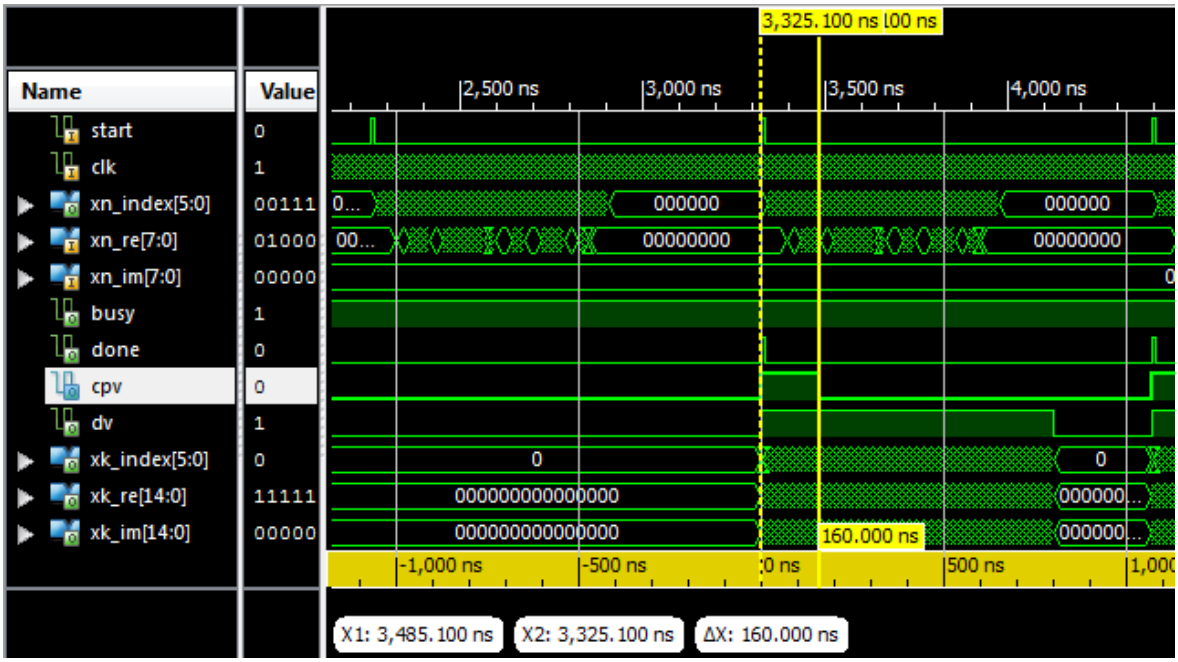

fig. 57 - Verificação do funcionamento do bloco IFFT_block(3)

Na figura 56 está ilustrado o tempo de um símbolo com o prefixo cíclico, e podemos ver que temos 80 ciclos de relógio, sendo que o relógio usado na simulação tem um período de 10 ns. Na figura 57 temos a duração do prefixo cíclico, 16 ciclos de relógio (160 ns), que corresponde a 1/4 do tempo da IFFT, que é de:

$T_{sym} - T_{GI} = T_{FFT} \rightarrow$ 800 - 160 = 640 ns.

4.5 Windowing

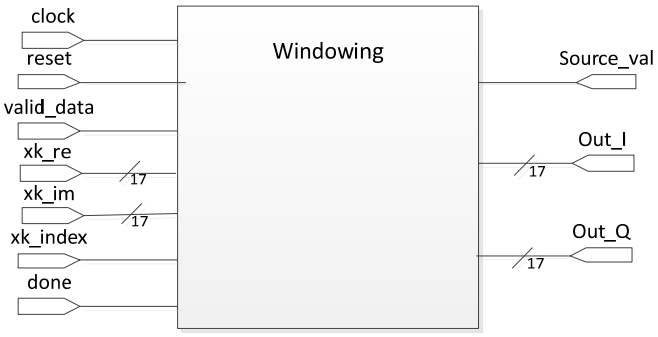

fig. 58 – Bloco Windowing

Tabela 18 – Interfaces do bloco Windowing

Sinais	Tipo	Descrição geral
Clock	Entrada	sinal de relógio do sistema
Reset	Entrada	Sinal de reset assíncrono do sistema
Valid_data	Entrada	Sinal que indica que os dados que provêm do bloco anterior são válidos
xk_re, xk_im	Entradas de 17 bits	Entradas de dados referentes aos dados reais e imaginários
xk_index	Entrada de 6 bits	Índice dos das amostras de entrada
done	Entrada	Sinal que indica o inicio de um símbolo
source_val	Saída	Sinal que indica ao bloco seguinte que os dados são válidos
out_I, out_Q	Saídas de 17 bits	Saidas de dados referentes aos dados reais e imaginários

Este bloco tem a função de fazer uma transição suave entre os símbolos OFDM. A função que este está a implementar é :

$$w_t[n] = \begin{Bmatrix} 1 & 1 \leq n \leq 79 \\ 0.5 & n = 0,80 \\ 0 & outros \end{Bmatrix}$$

Na prática passa por dividir a primeira e a ultima amostra de cada símbolo por dois. Isto está feito em VHDL, com um *shift* para a direita, tendo em conta depois do *shift* a salvaguarda do bit mais significativo, para que a amostra em complemento para dois não perca a informação de sinal. Para saber qual a primeira a mostra de cada símbolo este bloco aproveita o sinal de done gerado pela IFFT.

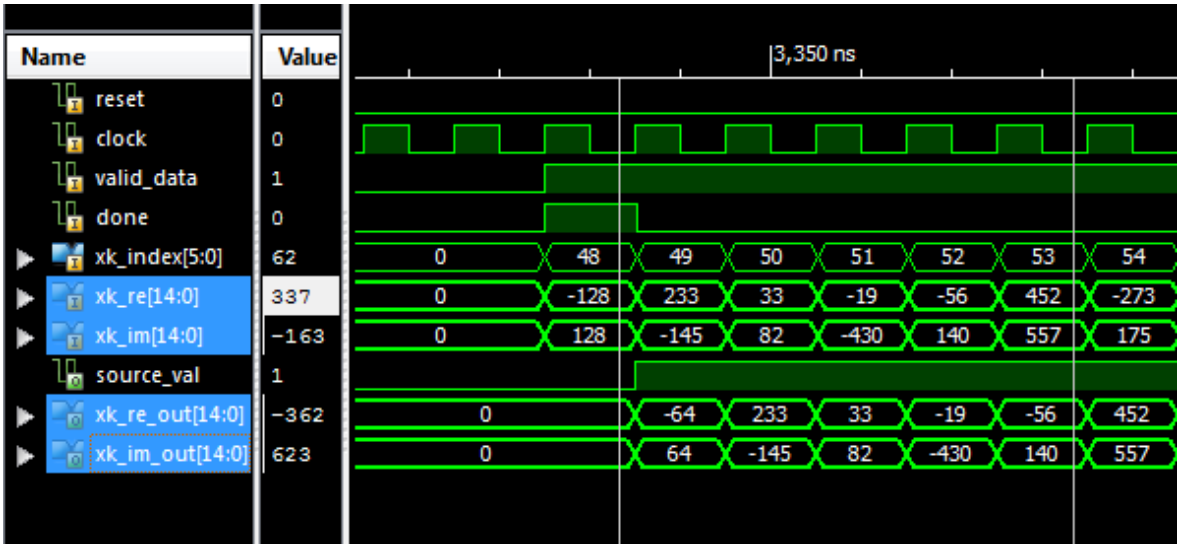

fig. 59 – Verificação do funcionamento do bloco Windowing

Na figura podemos ver as primeiras amostras de um símbolo na entrada do bloco, correspondentes às entradas xk_re e xk_im. Nas amostras de saída xk_re_out e xk_im_out, podemos ver a primeira amostra dividida.

4.6 Interface com as DACs

As DACs usadas neste projecto são de 10 bits complemento para 2. Os dados na saída da IFFT, têm 17 bits, portanto estes dados vão ter de sofrer um tratamento para poderem ser utilizados pelas DACs. A solução aplicada foi a de arredondar os valores de 17 bits, para valores de 10 bits. Para isto o bloco round_17_to_10 foi implementado em VHDL.

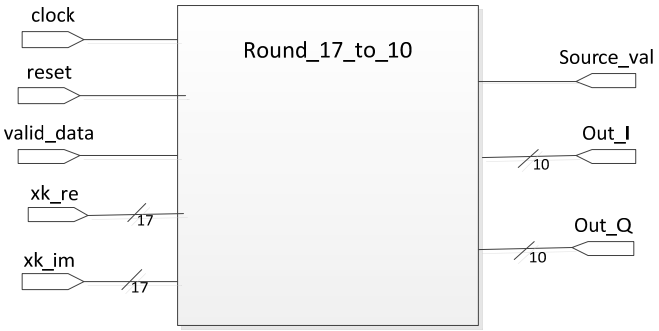

fig. 60 – Bloco round

Para fazer este arredondamento, verificou-se ao longo de uma trama aleatória qual o maior valor representado. Como esse valor precisa apenas de 14 bits para ser representado em complemento

para 2, permite que se descartem os 3 valores mais significativos, pois estes não contêm informação útil. Restam os 14 bits, que precisam de ser passados para apenas 10. Para estes os 4 bits menos significativos foram arredondados, e o valor do arredondamento somado aos 10 bits restantes, originando assim o resultado final com 10 bits, de acordo com as DACs.

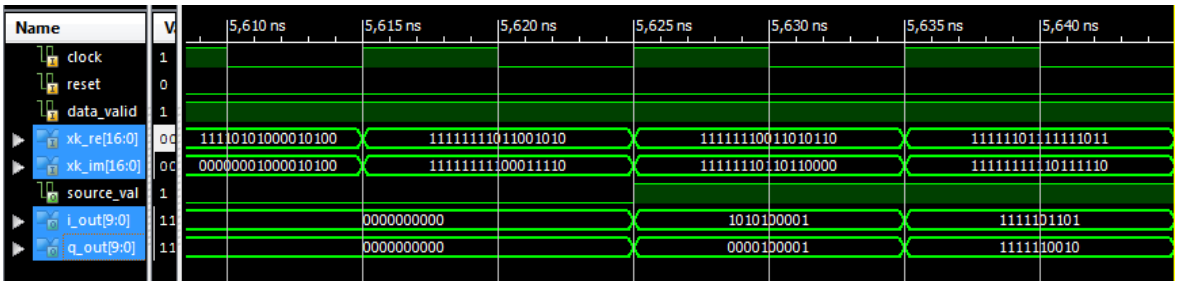

fig. 61 – Arredondamento executado pelo bloco round

Na figura acima pode ver-se os sinais de entrada com 17 bits, nas entradas xk_re e xk_im, e os correspondentes 10 bits nas saídas i_out e q_out respectivamente.

5 Recepção OFDM em 802.11p

5.1 Introdução

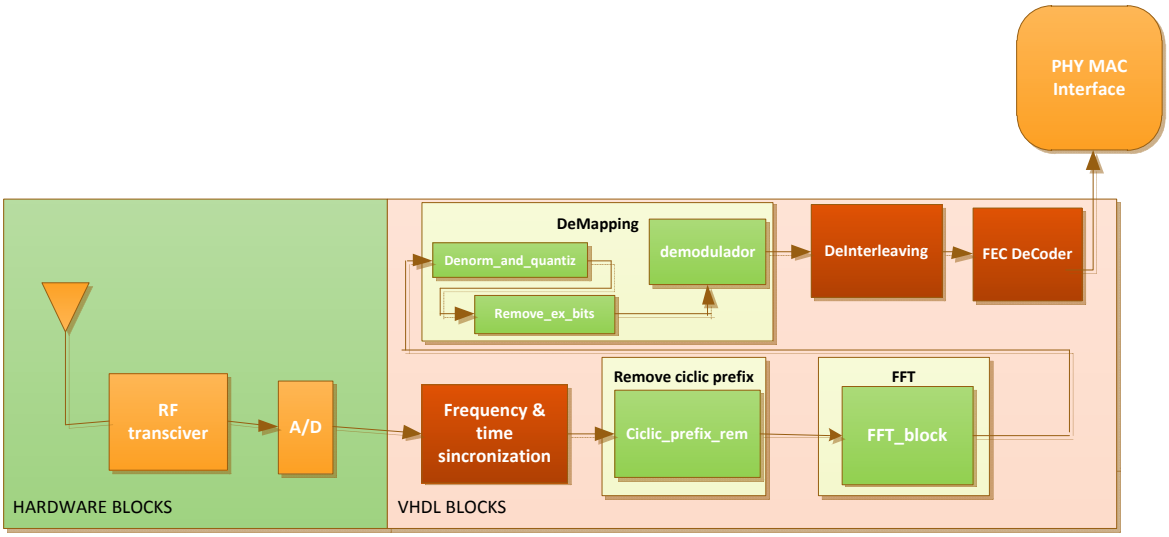

fig. 62 – Blocos da recepção

Como foi visto no terceiro capítulo a camada física do 802.11p tem a configuração de recepção que pode ser vista em esquema de blocos, na figura 62. No esquema de blocos estão representados os principais blocos constituintes da recepção e onde está especificado quais os que são implementados na FPGA. No mesmo esquema estão referenciados com cor verde, a parte dos blocos VHDL que foram alvo da minha implementação, e que vão ser descritos neste capítulo.

5.1.1 Abordagem de implementação

Para a implementação e validação da parte da recepção, foi utilizado um esquema semelhante, ao usado na transmissão. Por meio de uma test bench adequada, fazer a verificação individual de cada bloco, e depois de validado avançar para o seguinte. Tal como foi feito e se mostra na figura 33 do capitulo 4.

5.2 Remoção do Prefixo Cíclico

fig. 63 – Bloco Ciclic_prefix_remover

Tabela 19 –Interfaces do bloco cilic_prefix_remover

Sinais	Tipo	Descrição geral
Clock	Entrada	Sinal de relógio do sistema
Reset	Entrada	Sinal de reset assíncrono do sistema
Valid_data	Entrada	Sinal que indica que os dados provenientes do bloco antecessor são validos
Xk_re_in, xk_im_in	Entradas de 10 bits	Entradas de dados, referentes à parte real e imaginária
Start	Saída	Sinal que indica ao bloco de FFT, para começar a fazer os cálculos
Source_val	Saída	Sinal que indica ao bloco seguinte que os daos são válidos
out_I, out_Q	Siadas de 10 bits	Saídas de dados referentes à parte real e imaginária

Na recepção depois de feita a sincronização temporal e na frequência, estas operações são feitas essencialmente com recurso ao preambulo do início da trama (figura 31 do capítulo 3), deve passar-se do domínio do tempo e voltar ao domínio da frequência, para fazer a desmodulação da trama. Esta passagem para a frequência é feita fazendo a operação inversa da que foi feita na transmissão (IFFT), ou seja pela FFT. Para fazer a FFT, o primeiro passo a ser feito é o de retirar o prefixo cíclico inserido durante a transmissão. Este bloco tem a função de retirar o prefixo cíclico. Para além de retirar o prefixo cíclico este bloco faz o controlo do sinal de start da FFT, para esta começar a fazer os cálculos.

5.2.1 Validação comportamental do bloco ciclic_prefix_remover

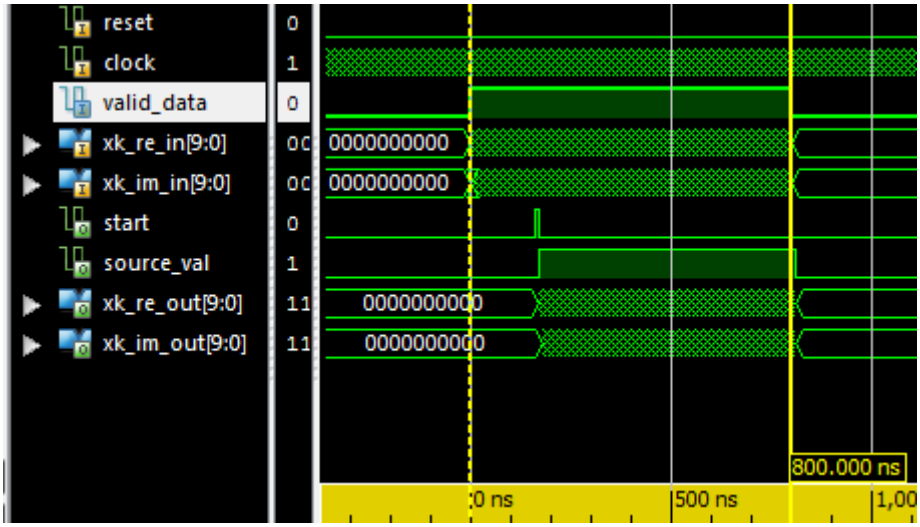

fig. 64 – Validação do comportamento do bloco cilcic_prefix_remover(1)

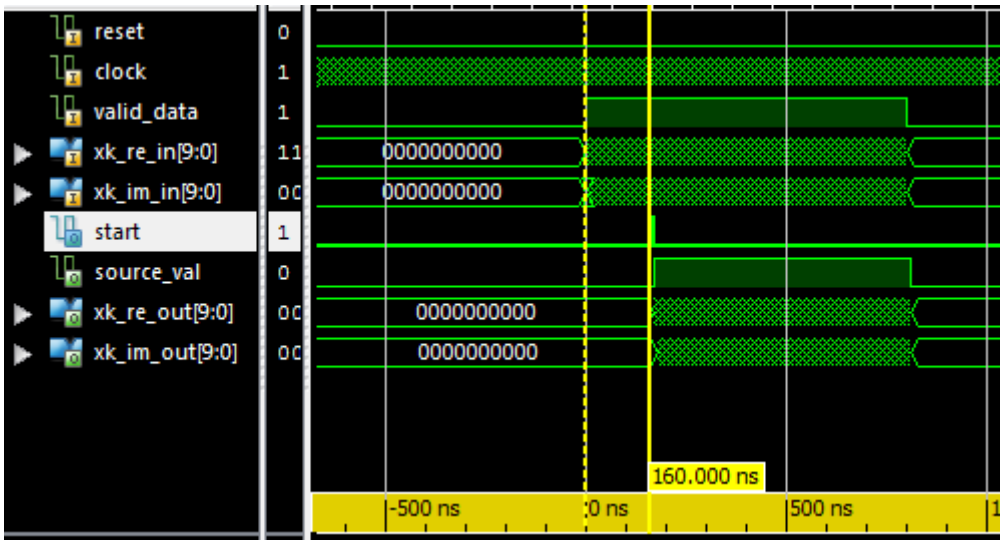

fig. 65 - Validação do comportamento do bloco cilcic_prefix_remover(2)

Nas figuras anteriores podemos ver um símbolo que chega a este bloco, e podemos verificar na saída deste bloco a remoção do prefixo cíclico. Podemos ver ainda a activação do sinal de start, depois de retirado o prefixo.

5.3 FFT

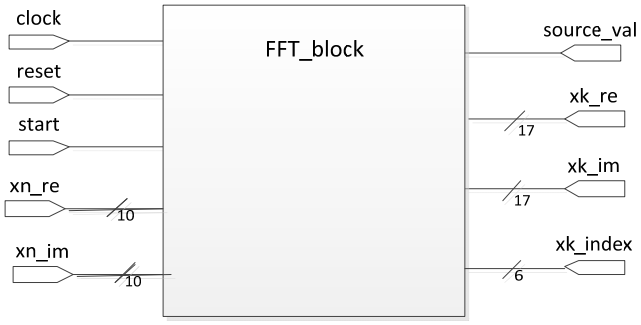

fig. 66 – Bloco FFT_block

Tabela 20 – Interfaces do bloco FFT_block

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
start	Entrada	Sinal que indica ao fft_core para iniciar os cálculos
xn_re, xn_im	Entradas de 10 bits	Entradas de dados, referentes à parte real e imaginária
xk_index	Saída de 6 bits	Sinal que indica o índice da amostra na saída
Source_val	Saída	Sinal que indica ao bloco seguinte que os dados são válidos
xk_re, xk_im	Saídas de 10 bits	Saídas de dados referentes à parte real e imaginária

$$x(k) = \sum_{n=0}^{N-1} x(k) e^{jnk\pi/N} \qquad , k = 0, \dots, N-1$$

Este bloco é essencialmente formado pelo mesmo IPcore usado na transmissão. As diferenças para o da transmissão encontra-se apenas no tipo de operação a realizar na não inserção do prefixo cilcico e como neste caso os dados de entrada não têm origem numa memória, a funcionalidade de três ciclos de relógio de atraso deixa de ser necessária. Todos os restantes parâmetros são semelhantes aos descritos para a IFFT, no capítulo anterior.

O controlo deste core passa apenas por definir o sinal fwd_inv a '1', bem como o seu *enable* fwd_inv_we. O restante controlo necessário é feito apenas pelo sinal de start, que é controlado no bloco anterior, o ciclic_prefix_remover. O esquema completo do bloco pode ser visto na figura abaixo.

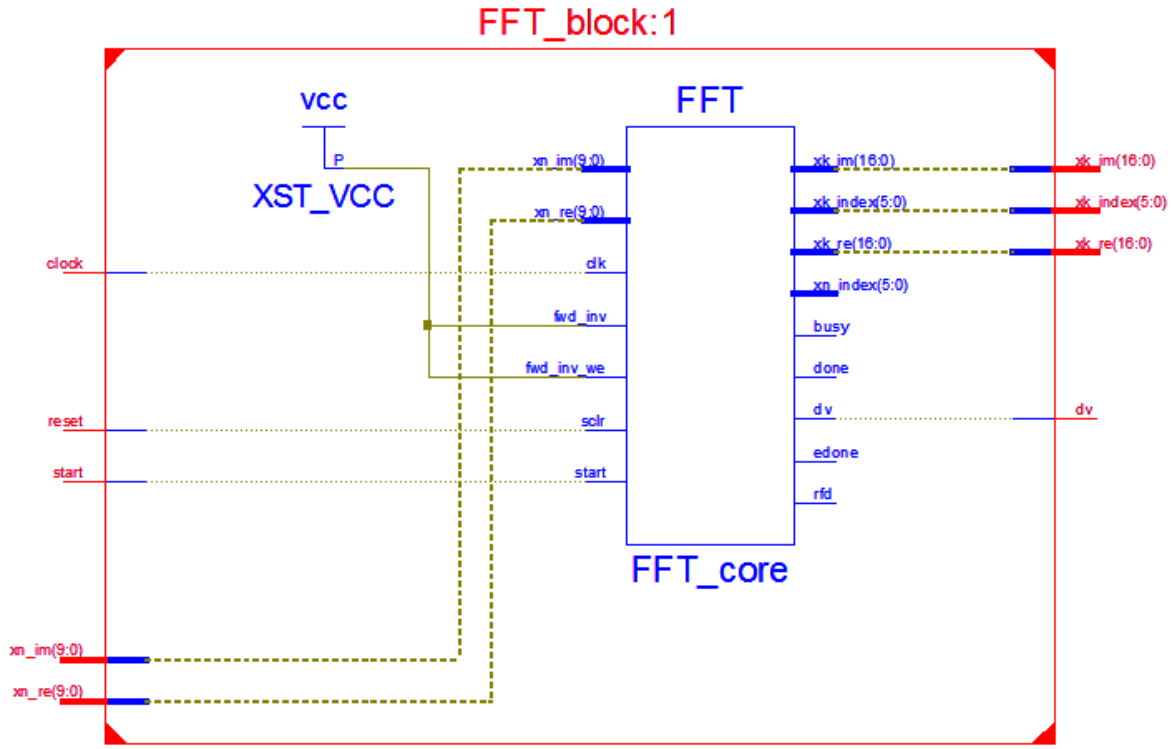

fig. 67 – Constituição interna do bloco FFT_block, retirado recorrendo à opção do ISE, RTL schematic que se encontra na parte de síntese do projecto.

5.3.1 Validação comportamental do bloco FFT_block

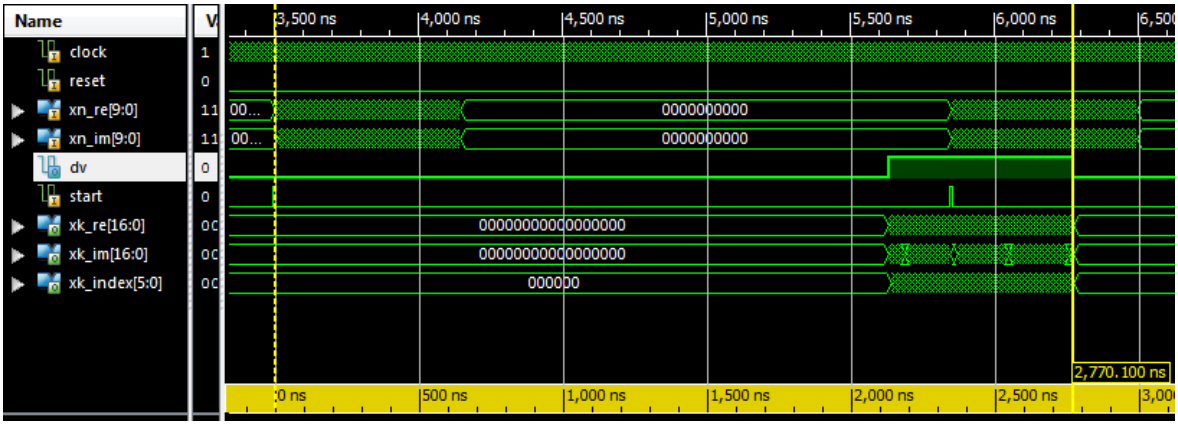

fig. 68 – Validação do comportamental bloco FFT_block

Na figura 68, pode ver-se a latência do bloco, neste caso são apresentados exactamente os 277 ciclos de relógio (2770 ns) entre o pulso de start até todos os dados terem sido calculados e apresentados na saída.

5.4 Desnormalização e Quantização

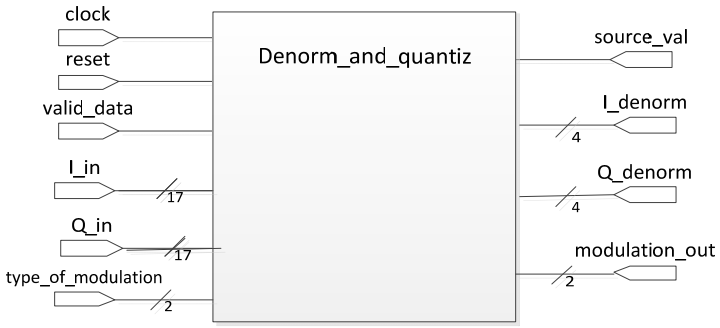

fig. 69 – Bloco Denorm_and_qantiz

Tabela 21 – Interfaces do bloco denorm_and_quantiz

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
valid_data	Entrada	Sinal que indica que os dados provenientes do bloco anterior são válidos
in_I, in_Q	Entradas de 17 bits	Entradas de dados, referentes à fase e quadratura
type_of_modulation	Entrada de 2 bits	Sinal que indica o tipo de modulação dos dados na entrada
source_val	Saída	Sinal que indica ao bloco seguinte que os dados são válidos
I_denorm, Q_denorm	Saídas de 4 bits	Saídas de dados referentes à fase e quadratura desnormalizados e quantizados.
modulation_out	Saída	Sinal que indica o tipo de modulação dos dados na saída

Uma vez que os dados sofreram uma normalização na transmissão, neste ponto da cadeia, eles encontram-se normalizados, e como tal para fazer a correcta desmodulação, estes devem ser desnormalizados. Este bloco tem então a função de fazer esta desnormalização dos dados provenientes da FFT. Mas nesta fase fazer a normalização, seria fazer a multiplicação pelo valor que depende da modulação a que se deu o nome de K_{DEMOD}.

Tabela 22 – Parametro dependente K_{DEMOD}

Modulação	K_{MOD}	$K_{DEMOD} = 1 / K_{MOD}$
BPSK	1	1
QPSK	$1/\sqrt{2}$	$\sqrt{2}$
16QAM	$1/\sqrt{10}$	$\sqrt{10}$
64QAM	$1/\sqrt{42}$	$\sqrt{42}$

$$d = \frac{I + jQ}{K_{MOD}} = (I + jQ)K_{DEMOD}$$

Por outro lado nesta fase, a recepção pode estar com erros, o que origina que a multiplicação por K_{DEMOD}, pode não dar exactamente os valores de amplitude das constelações. Por este facto é necessário fazer também uma quantização para colocar todos os valores em valores das constelações. Tal como na transmissão não foi necessária a utilização de multiplicadores para fazer a operação de normalização, aqui não se pode dizer que se pode fazer da mesma forma, pois os valores vindos da FFT, são próximos mas nunca iguais, de maneira a fazer uma entrada corresponder uma saída. Mas por outro lado a multiplicação não é necessária nesta fase, pois pode fazer-se apenas e logo a quantização. Ou seja, em vez de quantizar os dados depois de multiplicados, a quantização pode ser feita logo com os dados vindos da FFT. E é assim que este bloco está a operar. Ou seja analisando os valores provenientes da FFT para cada modulação, foram escolhidos os intervalos de quantização:

- BPSK – em BPSK, podemos dividir apenas os dados em positivos e negativos, assim:
  - Se I_in > 0, implica que estamos na presença de '1'.
  - Se I_in < 0, implica que estamos na presença de '-1'.

- QPSK – Em QPSK, temos a mesma situação que em BPSK replicada para a entrada I_in e Q_in.

- 16QAM – Neste caso são necessários intervalos, e os escolhidos foram:

  - '1' se está entre 0 e 500
  - '-1' se está entre 0 e -500
  - '3' se é maior que 500
  - '-3' se é menor que -500.

  Replicados os intervalos para I_in e Q_in.

- 64QAM – Tal como no anterior, neste caso são necessários intervalos, e os escolhidos foram:
    - ‘1’ se está entre 0 e 300
    - ‘-1’ se está entre 0 e -300
    - ‘3’ se está entre 300 e 600
    - ‘-3’ se está entre -300 e -600
    - ‘5’ se está entre 600 e 900
    - ‘-5’ se está entre -600 e -900
    - ‘7’ se é maior que 900.
    - ‘–7’ se é menor que -900.

  Replicados os intervalos para I_in e Q_in

5.4.1 Validação comportamental do bloco Denorm_and_quantiz

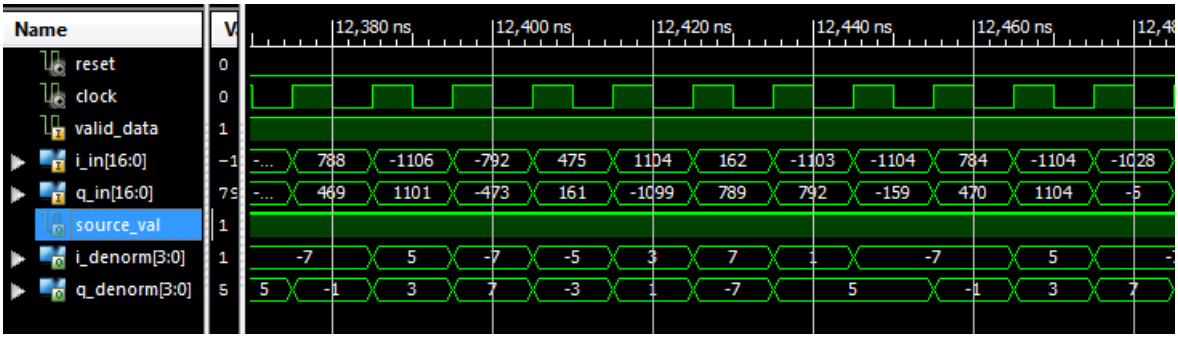

fig. 70 – Validação comportamental do bloco Denorm_and_quantiz

Na figura pode-se ver os valores, convertidos par decimal para uma mais fácil visualização, dos sinais de entrada (os valores que vêm da FFT), e os sinais de saída quantizados. Nesta simulação estamos a ver um símbolo que foi modulado em 64QAM, e portanto os seus valores para a constelação são ±1, ±3, ±5, ±7, e como se pode ver nesta pequena amostra do símbolo, todos os valores de saída estão coincidentes com os valores da constelação.

5.5 Remoção das portadoras extra

fig. 71 – Bloco remove_ex_bits

Este bloco serve para retirar as portadoras adicionadas na transmissão que não contêm dados. Vai retirar as portadoras piloto, e as portadoras não utilizadas, de modo a ter na saída do bloco apenas as 48 portadoras de dados, que vão ser desmoduladas no bloco seguinte. Para além disto devido ao mapeamento especifico das portadoras na IFFT, temos que neste ponto estas se encontram nesse mapeamento. Assim este bloco tem também a função de voltar a colocar as portadoras nas posições originais de modo a poderem ser correctamente demepeadas nos bits originais. Recapitulando o que foi dito no capitulo anterior, as portadoras 1 a 27 estão nas mesmas posições na IFFT, logo estas são as primeiras que aparecem nas entradas deste bloco. Analisando a figura 44 estas portadoras são as últimas que devem aparecer nas saídas deste bloco. Assim as portadoras 1 a 27 de dados são armazenadas num vector auxiliar, sendo as posições correspondentes às portadoras descartadas neste ponto. De seguida temos as portadoras nulas que também são descartadas, até que se chega à posição 38 da saída da FFT onde se encontram, a partir desta posição, as portadoras -26 a -1 que são as primeiras a aparecer segundo a figura 44. Como estas são as primeiras que devem aparecer estas portadoras são colocadas directamente na saída, sendo excluídas obviamente as posições contendo as portadoras piloto. Quando chegado à posição 63 (portadora -1), as portadoras previamente armazenadas no vector auxiliar, são colocadas por ordem na saída. E este processo repete-se para todos os símbolos.

5.5.1 Validação comportamental do bloco remove_ex_bits

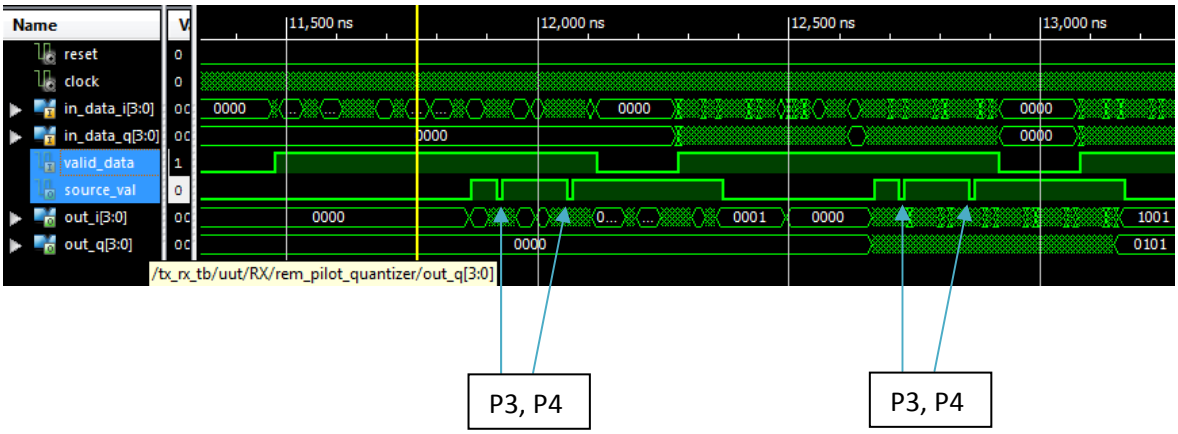

fig. 72 – Validação Comportamental do bloco remove_ex_bits

Na figura 72, temos representado os símbolos na entrada do bloco, com as 64 portadoras, e podemos ver na saída as 48 portadoras de dados podendo notar-se as duas piloto que retiradas. As restantes duas não se vêem pois estas portadoras já foram descartadas e não foram armazenadas no vector, sendo apenas visível as portadoras de dados. Tudo isto pode ser visto recorrendo ao sinal de valid_data para a entrada, e o sinal source_val para saída.

5.6 Desmapeamento

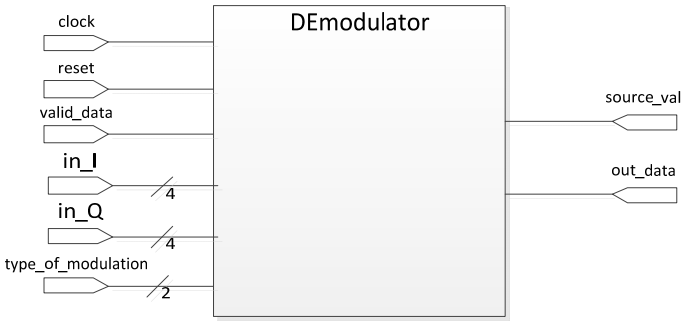

fig. 73 – Bloco DEmodulator

Tabela 23 – Interfaces do bloco remove_ex_bits

Sinais	Tipo	Descrição geral
clock	Entrada	Sinal de relógio do sistema
reset	Entrada	Sinal de reset assíncrono do sistema
valid_data	Entrada	Sinal que indica que os dados provenientes do bloco anterior são válidos
in_I, in_Q	Entradas de 4 bits	Entradas de dados, referentes à fase e quadratura
type_of_modulation	Entrada de 2 bits	Sinal que indica o tipo de modulação dos dados
source_val	Saída	Sinal que indica ao bloco seguinte que os dados são válidos
out_data	Saída	Sida da stream de bits desmodulados

Este é o último bloco que se apresenta, pois com este bloco termina a parte da cadeia que me coube trabalhar.

Na entrada do bloco, a cada amostra corresponde um ciclo de relógio, mas dependendo do tipo de modulação, na saída temos para cada amostra de entrada, 1, 2, 4 ou 6 ciclos de relógio, dependendo da modulação (BPSK, QPSK, 16QAM e 64QAM). Como os ritmos de da entrada e saída são diferentes, isto implica a utilização de um fifo na entrada. O fifo tem a função de guardar as novas amostras até que o processo de desmodulação da amostra anterior seja concluído. Assim o bloco DEmodulator, é um bloco que engloba o fifo de entrada e o bloco que executa a função da desmodulação, como se pode ver na figura seguinte.

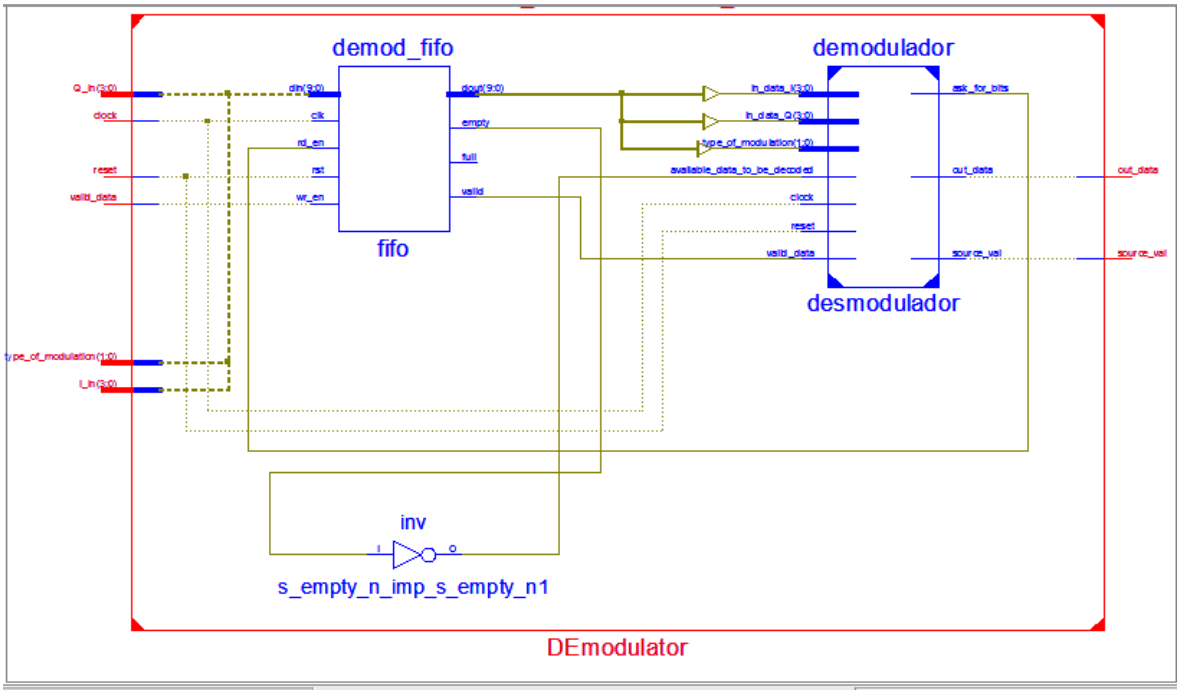

fig. 74 – Esquema interno do bloco DEmodulator

O tamanho do fifo tem de ser tal que possa guardar grande parte dos símbolos da trama. Como para estes testes apenas se vão utilizar tramas não muito grandes (200 Bytes máximo), o tamanho deste fifo é de 10 x 512. A largura de dez é devido a que os bits 10 a 6 do barramento de dados do fifo conterem os dados da fase, 5 a 2 os dados da quadratura, e por fim os bits 1 a 0 contêm a modulação em que estes dados se encontram, informação esta que vem do bloco anterior. A informação sobre a modulação é importante devido ao campo SIGNAL, e já foi averiguada anteriormente.

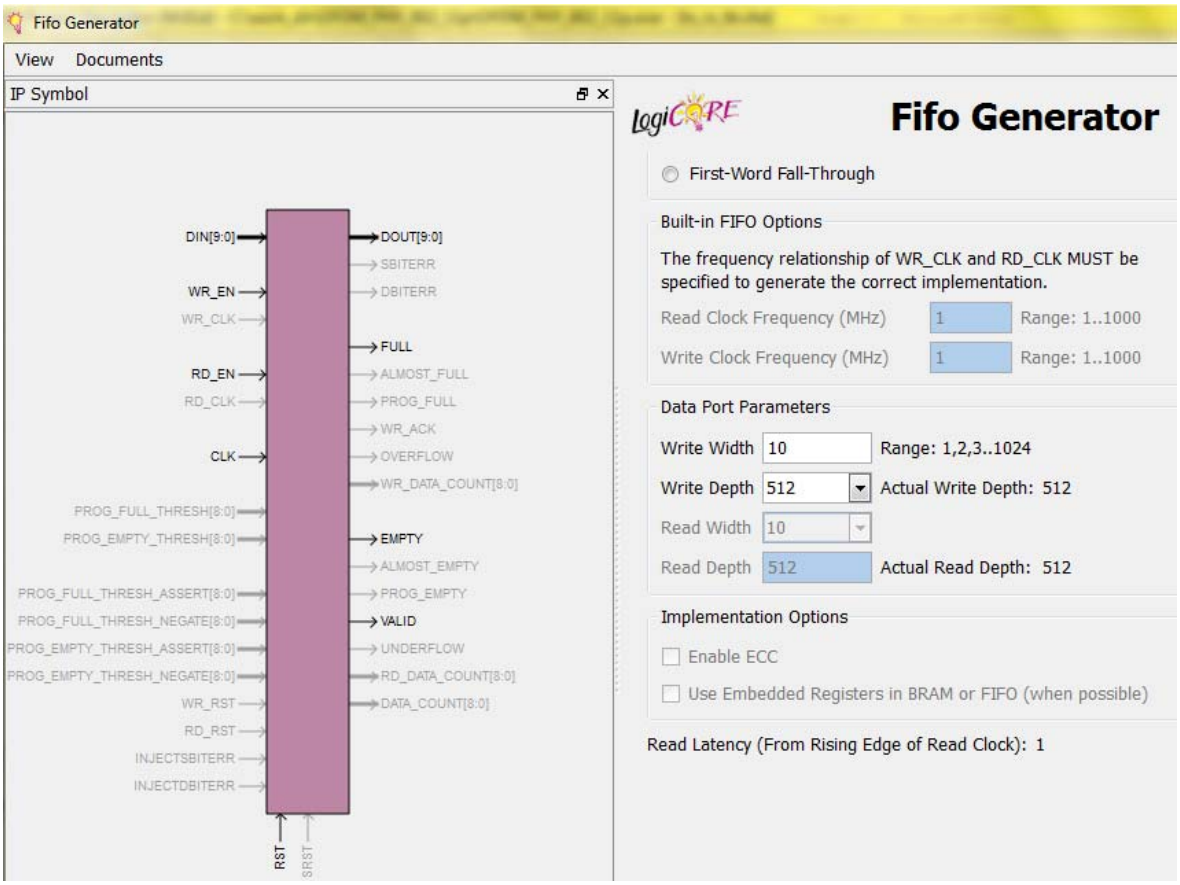

fig. 75 – Parâmetros do fifo do bloco DEmodulator

5.6.1 Validação comportamental do bloco DEmodulator

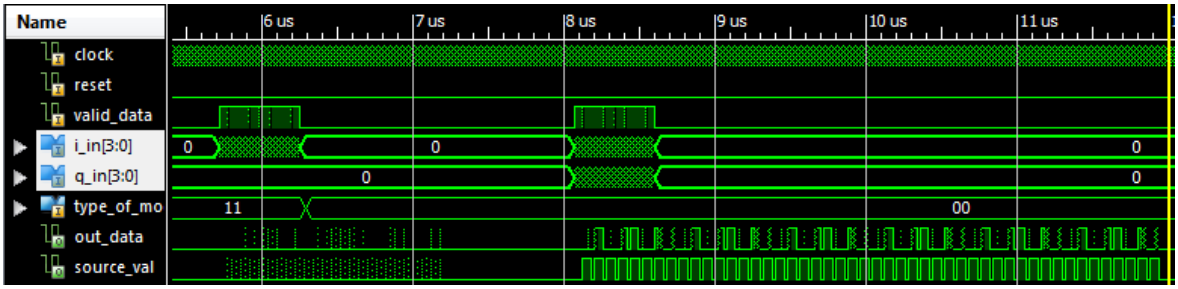

fig. 76 – Validação comportamental do bloco DEmodulator (1)

Na figura 76 identifica claramente a utilidade do fifo. Se olharmos para os sinais de entrada valid_data, bem como os sinais i_in e q_in, vemos que estes chegam num período de aproximadamente 0.5 µs (sendo que o período do relógio na simulação é de 10 ns). Corresponde à duração das 50 portadoras. Comparando com os dados desmodulados na saída out_data e source_val, vemos que a duração dos dados desmodulados é bastante maior, o que implica a utilização do fifo. Pode ver-se também uma desvantagem que este coloca, pois para a correcta desmodulação o processo de desmodulação pede dados ao fifo quando está pronto para receber uma nova amostra. Desde que é pedida a nova amostra os dados apenas ficam válidos na saída do fifo um ciclo de relógio depois. Isto provoca uma latência de dois ciclos de relógio entre cada amostra que é pedida pelo processo de desmodulação.

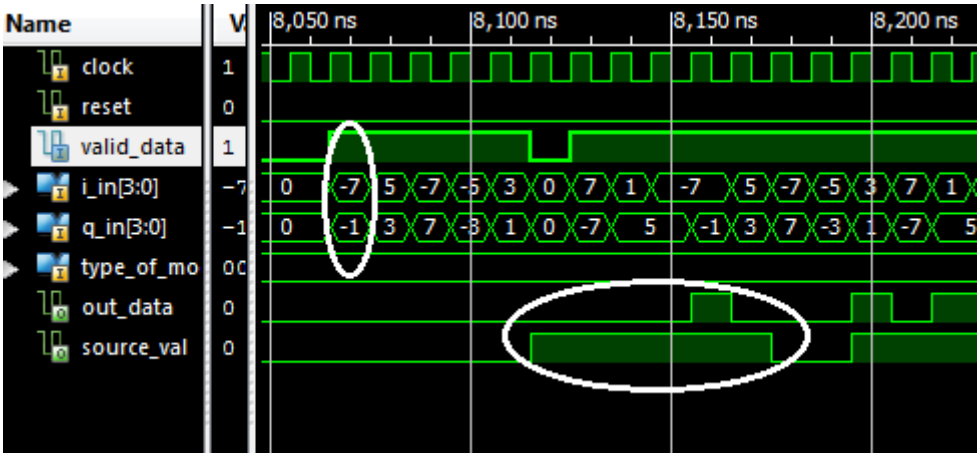

fig. 77 - Validação comportamental do bloco DEmodulator (2)

Na figura 77, é pretendido mostrar um exemplo da correcta desmodulação feita pelo bloco, neste caso numa modulação 64QAM.

Tabela 24 – bits de desmodulação 64QAM

Bits de saída $b_0b_1b_2$	I	Bits de saída $b_3b_4b_5$	Q
000	-7	000	-7
001	-5	001	-5
011	-3	011	-3
010	-1	010	-1
110	1	110	1
111	3	111	3
101	5	101	5
100	7	100	7

Com a ajuda da tabela 24 podemos ver que I = -7 e Q = -1, corresponde a uma sequência de saída de '000010', o que corresponde ao que está na figura.

6 Simulação da cadeia completa e testes em hardware

O hardware utilizado no trabalho foi o Kit NEXYS2 da *Digilent*, que tem como principal componente uma FPGA SPARTAN 3E 500. Este kit para além da FPGA fornece, um relógio de 50MHz, vários botões e interruptores, vários Leds, 4 leds de 7 segmentos, entre outros dispositivos de I/O. Com estes componentes instalados, torna-se mais fácil fazer o *debug* do projecto na placa. Os botões e interruptores podem, por exemplo, simular entradas enquanto que os leds podem servir como o indicador visual das saídas.

Para além disto a *Digilent* disponibiliza um *plugin*, que permite a utilização do *ChipScope* sem utilizar o cabo JTAG. Com este *plugin* consegue-se com o cabo USB que liga a placa ao computador, fazer *debug* sem o cabo JTAG, o que se mostrou uma grande vantagem, pois o cabo disponibilizado para este trabalho era partilhado com outras pessoas, e deste modo pude prescindir dele. A desvantagem deste Kit é a pequena FPGA que possui, que não permite o teste em hardware de toda a cadeia (transmissão e recepção) com o *ChipScope*.

6.1 Simulação da cadeia completa

Para o testar toda a cadeia, transmissão e recepção, o teste que foi feito foi o de ligar directamente a cadeia de transmissão à cadeia de recepção. Este teste consiste basicamente em colocar uma trama no inicio da cadeia de transmissão, e ver o resultado no final da cadeia de recepção. Se tudo se estiver a comportar correctamente a trama inicial e a final devem ser iguais. Com este teste pode ver-se também toda a latência que estes blocos provocam na camada física.

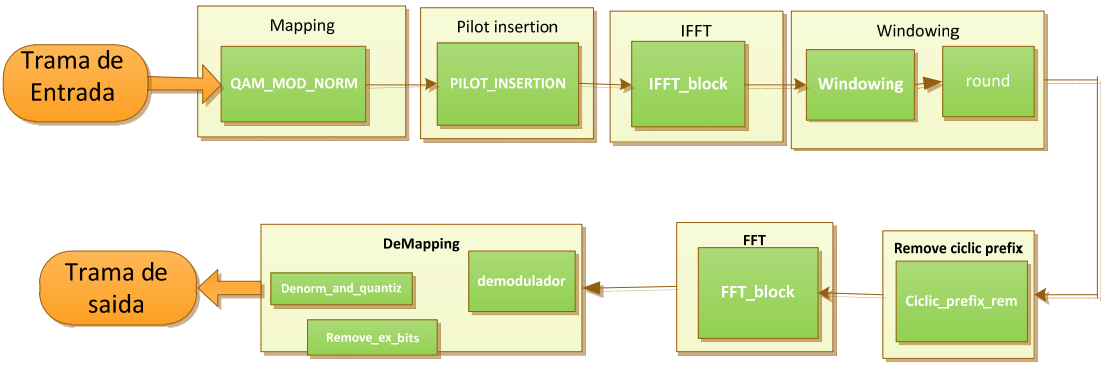

fig. 78 –Esquema do teste a toda a cadeia

Na *test bench* realizada para o teste, foi incluído um processo para armazenar os bits descodificados num vector, para que depois esse vector possa ser comparado com o vector de bits colocados na entrada. A Bit *stream* de entrada tem um total de 336 bits para poder ser visualizada completamente na simulação, e ao mesmo tempo ter o tamanho suficiente para acomodar a modulação em 64QAM, e para isto este é o valor mínimo.

Nas figuras seguintes são apresentados os resultados.

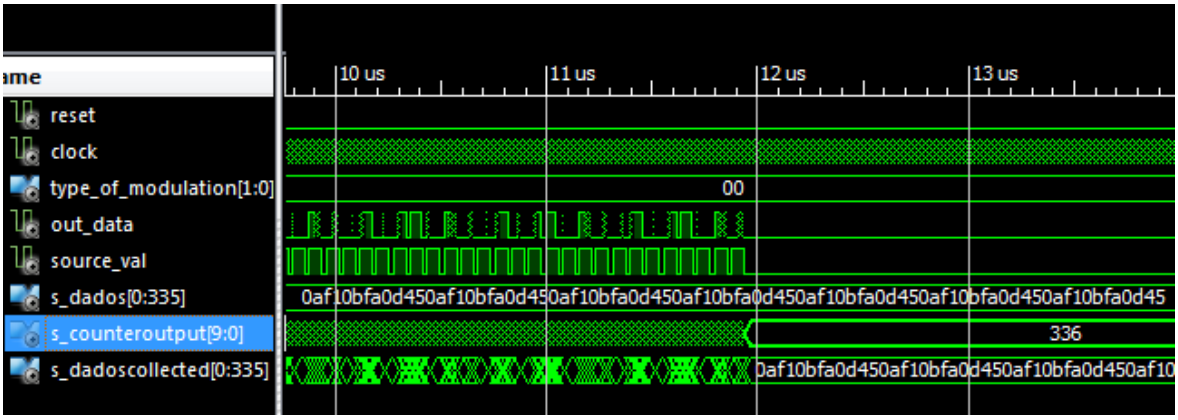

fig. 79 – Final da bit stream colectada.

Na figura 79 pode ser visto o final da trama, e podemos ver no sinal seleccionado, que foram capturados 336 bits. Na imagem seguinte apresenta-se a totalidade das duas tramas para poder ser feita a comparação. O vector s_dados é bit *stream* de entrada e o s_dadoscollected os bits capturados na saída out_data.

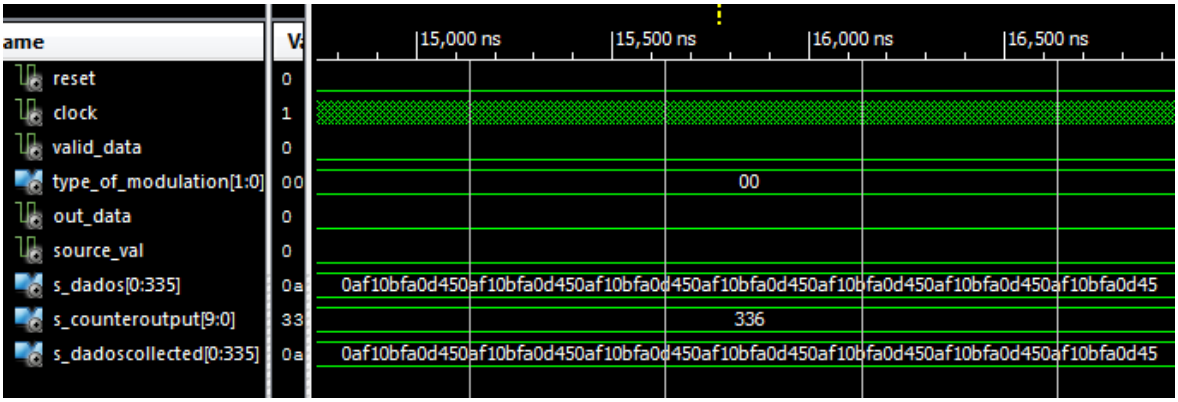

fig. 80 – Trama de entrada e trama capturada

Podemos ver, comparando as duas tramas, que estas são iguais, o que mostra o correcto funcionamento de toda a cadeia. No caso desta simulação temos uma modulação feita em 64QAM, mas foi verificado para todas as outras.

Pode ser visto nesta simulação também o tempo total de latência introduzido por todos os blocos da transmissão e recepção. Obteve-se um resultado de 5.730 µs, sendo que o clock da simulação é de 10 ns, o que implica uma latência de 573 ciclos de relógio. Estes 573 ciclos dividem-se em 330 desde que temos o primeiro bit na entrada da cadeia de transmissão até que o primeiro esteja pronto para ir para a DAC. Os restantes 243 são a latência da recepção.

6.2 Teste em Hardware

Com a ajuda do *ChipScope*, uma ferramenta disponibilizada pela XILINX juntamente com o ISE, é possível ver os sinais no hardware real. É possível carregar o programa na FPGA, com hardware extra ao programa pertencente ao *ChipScope*, que permite a visualização e validação ou identificação de erros que ocorram no hardware. Para isto é apenas necessário adicionar no ISE, um novo *chipScope Definition and connection file* ao projecto, onde se define os sinais a capturar e a quantidade de dados que vai ser capturada. Como se imagina este processo requer recursos extra da FPGA, nomeadamente memória. Assim o programa não deve ocupar toda a FPGA, o que está praticamente a acontecer como pode ser visto na figura 81, para que esta possa acomodar os recursos extra do *ChipScope*. Um exemplo da quantidade de recursos é apresentado na figura 81.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	4424	4656	95%
Number of Slice Flip Flops	5938	9312	63%
Number of 4 input LUTs	7909	9312	84%
Number of bonded IOBs	50	232	21%
Number of BRAMs	4	20	20%
Number of MULT18X18SIOs	12	20	60%
Number of GCLKs	1	24	4%

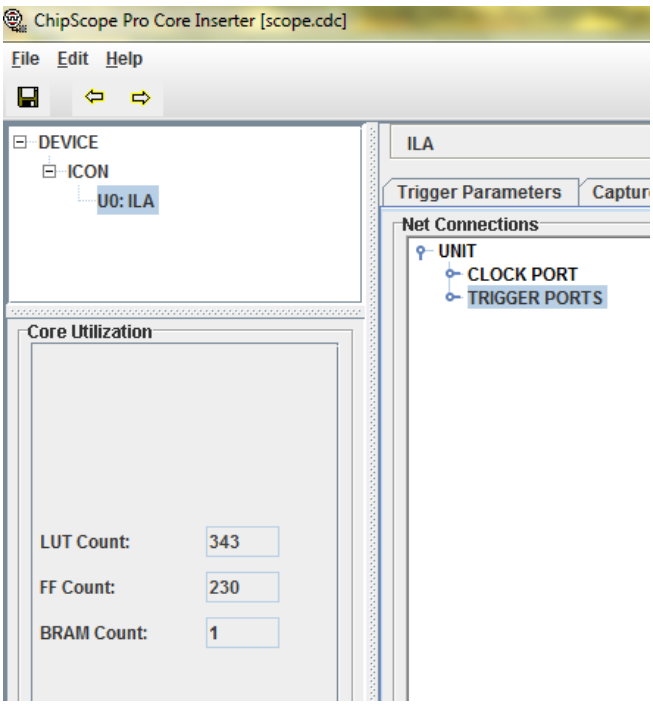

fig. 81 – Exemplo de recursos usados pelo projecto e os recursos extra utilizados pelo *ChipScope*.

Posto isto, para validar a cadeia no hardware a abordagem foi comparar os resultados obtidos na simulação, que já foram validados nos pontos anteriores, com os resultados obtidos com o *ChipScope*.

Para a realização dos testes no hardware, os dados de entrada que eram definidos na *test bench*, agora não o podem ser. Para a resolução desta questão implementou-se uma memória com a trama e esta é lida fornecendo assim os dados de entrada da cadeia.

6.2.1 Validação da transmissão em Hardware

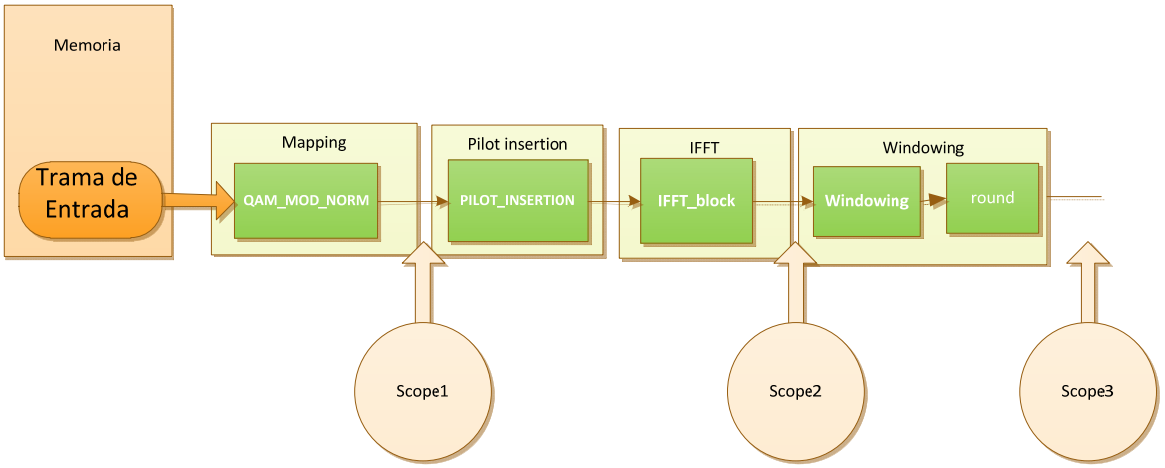

fig. 82 – Ilustração das análises com o *ChipScope*.

Podemos ver na figura anterior os pontos do circuito que foram vistos com o *ChipScope*, e comparados com os resultados em *test bench*, que se apresentam de seguida. Estes pontos são definidos pelos sinais que se escolhem para ver com o *ChipScope*. De referir também que as capturas feitas pelo *ChipScope*, ou seja as amostras capturadas, são capturadas quando o sinal de source_val no ponto é activo. O sinal de source_val é o sinal de *trigger* para a captura no ponto *Scope*, ou seja na primeira transição de '0' para '1' deste sinal, inicia a captura até encher os buffers definidos para armazenamento dos dados capturados.

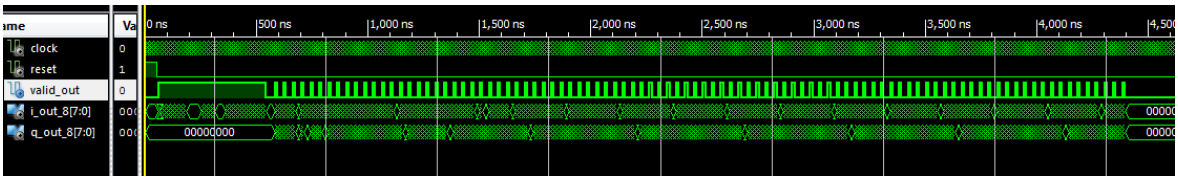

fig. 83 – Vista de toda a trama no ponto do scope1 na simulação.

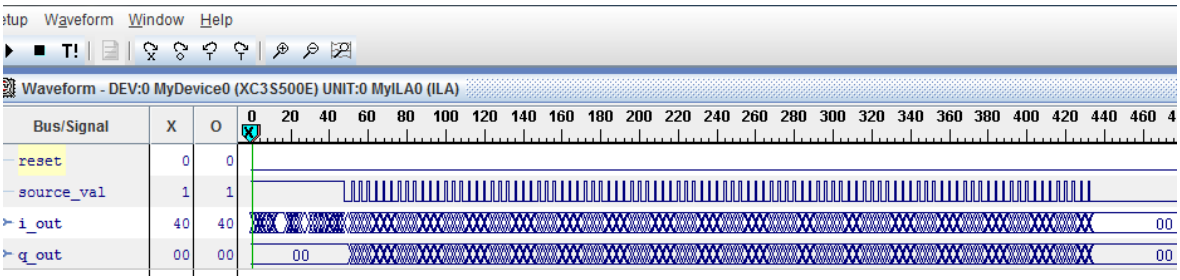

fig. 84 - Vista de toda a trama no ponto do scope1 no *ChipScope*.

Analisando as duas imagens anteriores podemos ver que o comportamento em simulação é semelhante ao que se passa no hardware. Para confirmar em termos de valores apresenta-se de seguida um zoom nestas imagens:

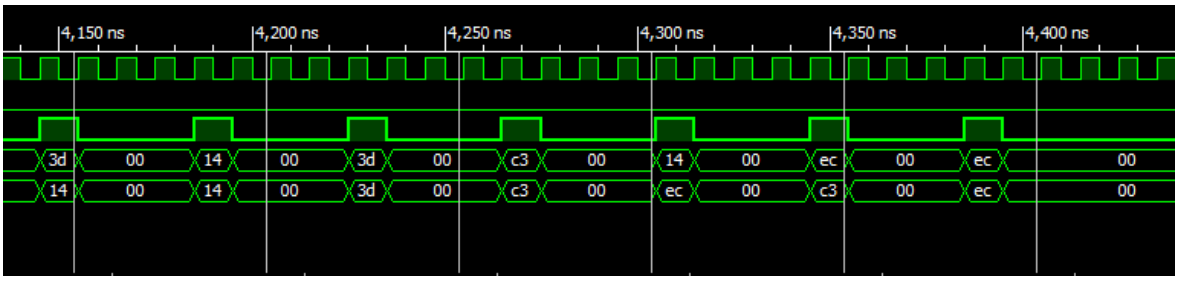

fig. 85 - Vista zoom no ponto do scope1 na simulação.

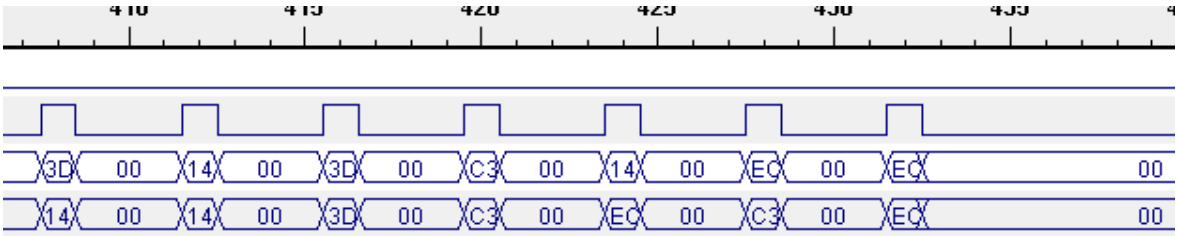

fig. 86 - Vista zoom no ponto do scope1 no *ChipScope*.

Visto o zoom das imagens podemos ver que temos a mesma coisa em simulação e em hardware, ou seja na saída do bloco de *mapping*, tudo está a funcionar correctamente. Vamos avançar para o ponto seguinte.

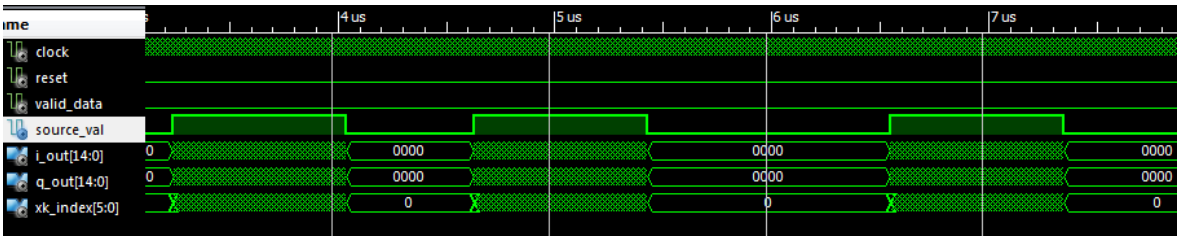

fig. 87 - Vista de toda a trama no ponto do scope2 na simulação.

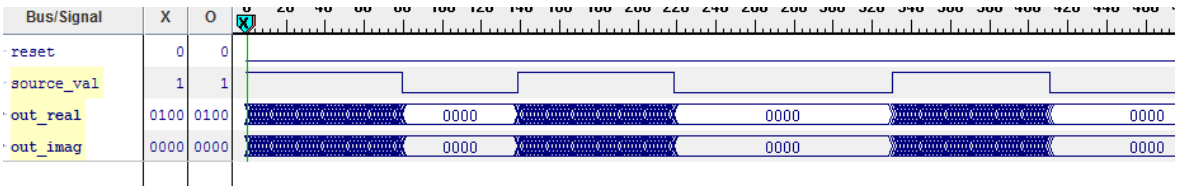

fig. 88 - Vista de toda a trama no ponto do scope2 no *ChipScope*.

No ponto scope2, saída da IFFT podemos ver que o comportamento em hardware é semelhante ao que se passa em simulação. Para uma verificação dos valores mostra-se novamente um zoom.

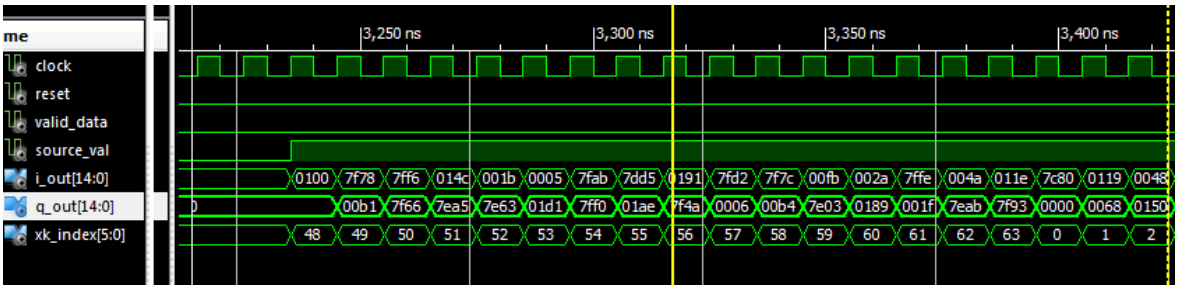

fig. 89 - Vista zoom no ponto do scope2 na simulação.

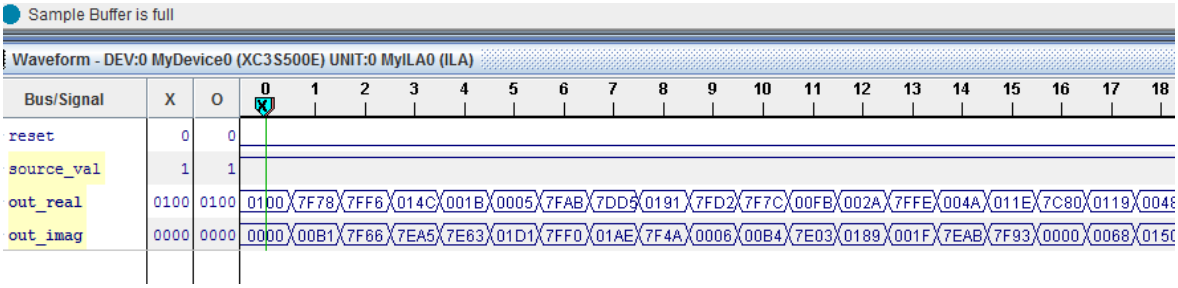

fig. 90 - Vista zoom no ponto do scope2 no *ChipScope*.

Com a vista de zoom podemos ver que em termos de valores as análises também coincidem, o que mostra que até à saída da IFFT, tudo se está a comportar correctamente no hardware. Por fim o último ponto analisado vai mostrar as saídas do sistema, e novamente vai ser feita a comparação com a simulação.

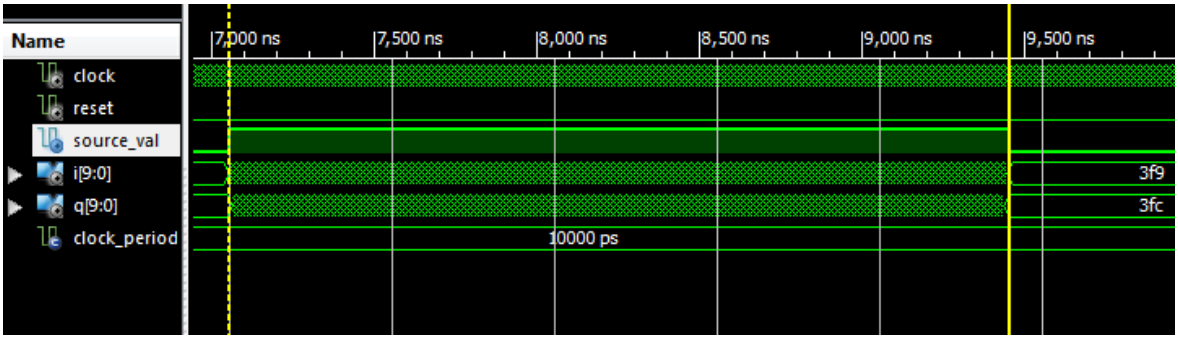

fig. 91 - Vista de toda a trama no ponto do scope3 na simulação.

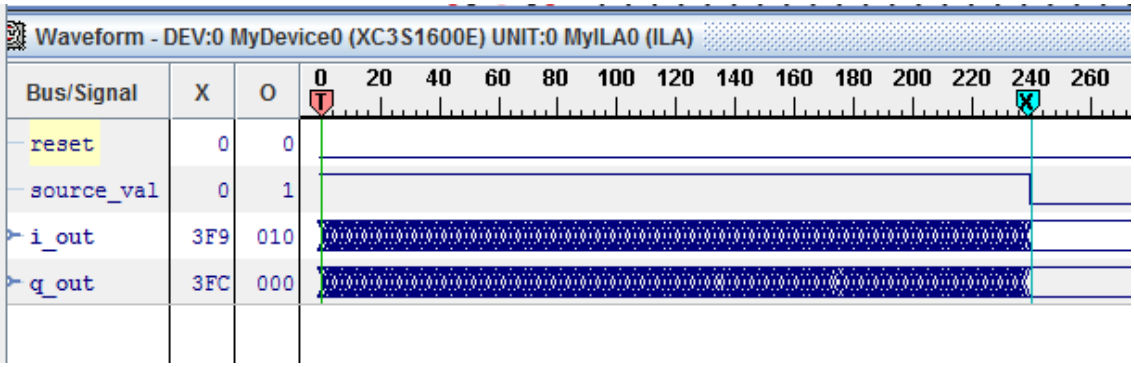

fig. 92 - Vista de toda a trama no ponto do scope3 no *ChipScope*.

Neste ponto podemos ver toda a duração da trama, pronta a ser transmitida para o meio. Podemos ver nas figuras que os comportamentos em hardware e em simulação coincidem. Em termos nominais mostra-se o zoom destas imagens de seguida.

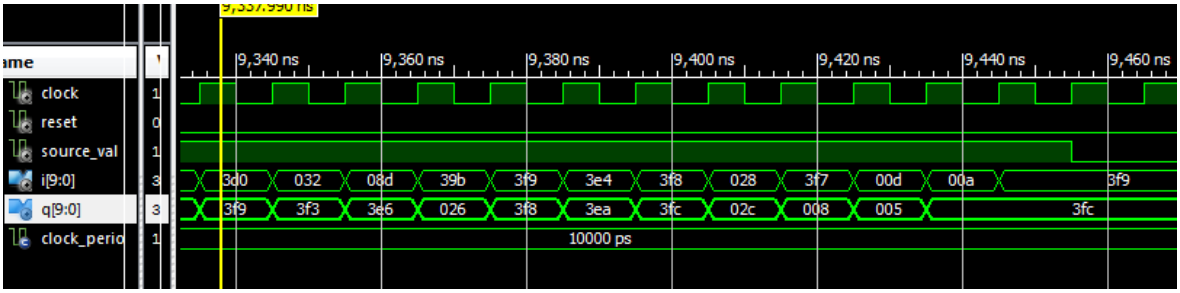

fig. 93 - Vista zoom no ponto do scope3 na simulação.

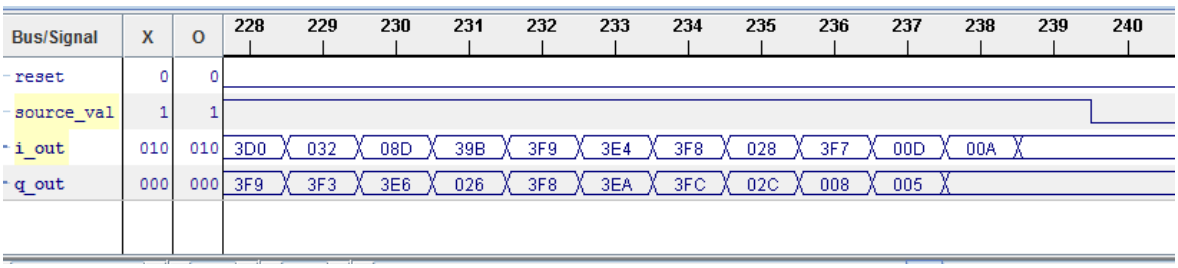

fig. 94 - Vista zoom no ponto do scope3 no *ChipScope*.

Assim depois de verificados alguns pontos intermédios, e o ponto final na cadeia de transmissão, pode-se afirmar que a cadeia de transmissão está a funcionar correctamente no hardware.

6.2.2 Visualização de trama transmitida no VSA

Para validar a transmissão foi realizado um teste que consiste no processamento pela cadeia de transmissão, de uma trama completa 802.11p. Este teste consistiu então em capturar os valores das saídas *In-phase* e *Quadrature,* no final dos meus blocos, ou seja os bits que vão ser convertidos pelas DACs. Estes dados capturados em simulação, foram introduzidos numa memória, que como o sistema completo de transmissão ainda não está, neste ponto do projecto HEADWAY, implementado na FPGA, a memória vai servir como a simulação da cadeia implementada. Esta memória vai descarregar os dados capturados para as DACs, e o restante sistema de Hardware que já se encontra implementado, e é feita uma análise com recurso ao aparelho de medida (VSA), que tem a capacidade de capturar e descodificar a norma 802.11a e j, que na sua camada física são semelhantes ao 802.11p. De seguida mostra-se o resultado visualizado.

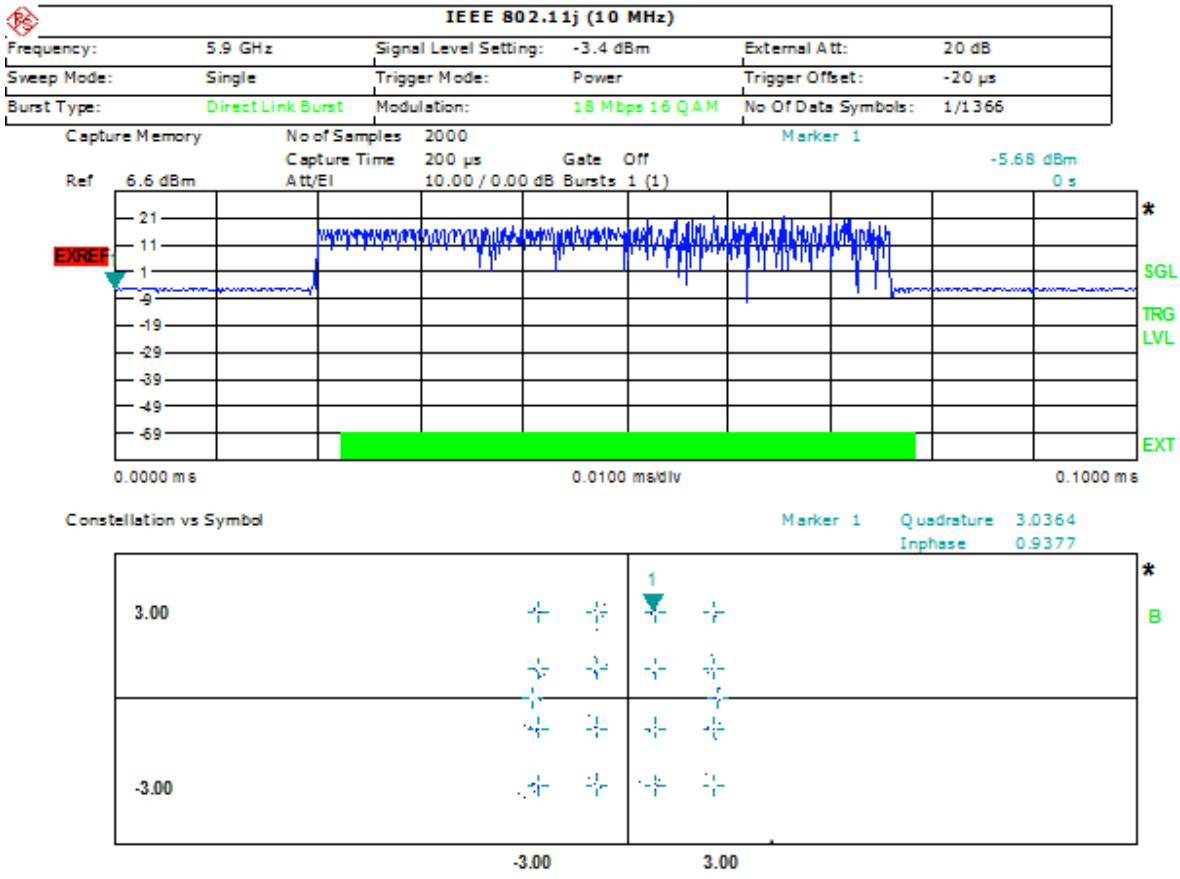

fig. 95 – Captura da trama OFDM 802.11p, pelo VSA.

Pode ser vista na figura no primeiro gráfico, temos o espectro da trama enviada. No segundo gráfico temos a constelação 16QAM, e os pontos descodificados pelo VSA (em zoom na imagem seguinte).

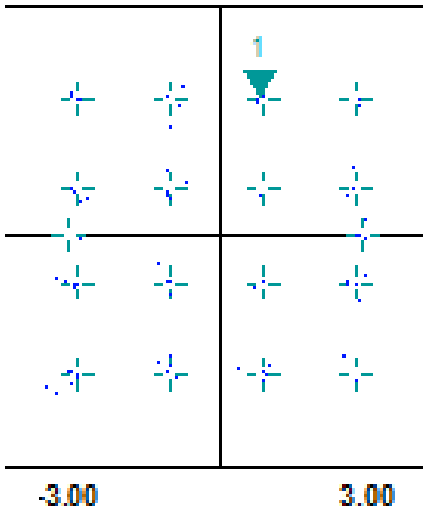

fig. 96 – Captura da constelação da trama enviada

6.2.3 Validação da recepção em Hardware

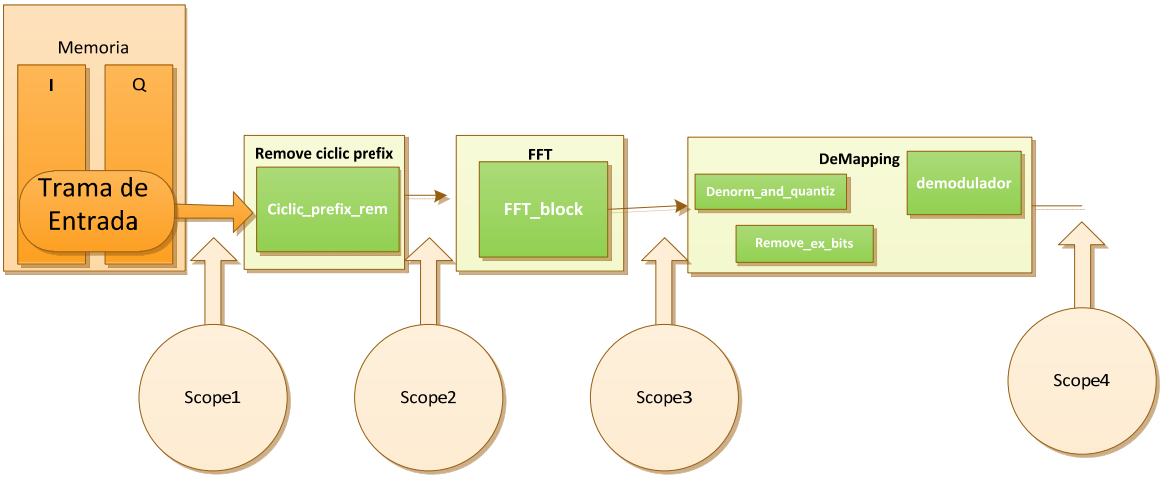

fig. 97 – Ilustração das análises com *ChipScope* na recepção.

Para a validação da recepção, foi usado o mesmo processo que na transmissão e que foi apresentado no ponto anterior. Para este teste a memória contendo a trama a ser processada pela cadeia, é na verdade um conjunto de duas memórias, uma contendo o conteúdo da fase de

uma transmissão, e a outra o conteúdo da quadratura. Portanto o primeiro ponto analisado com o *ChipScope* é precisamente a saída da memória, de maneira a validar a correcta inserção dos dados na cadeia.

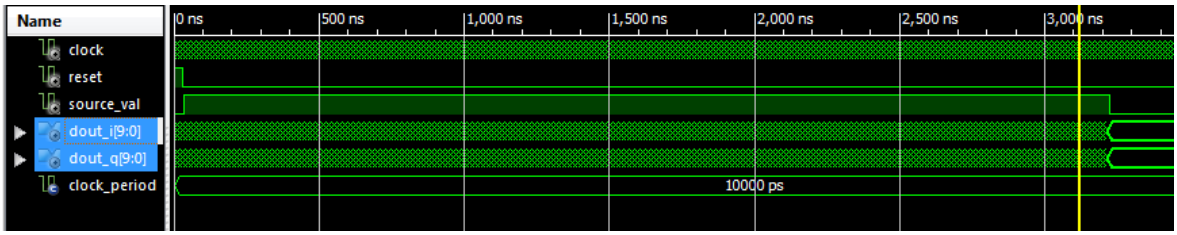

fig. 98 – Saída da memória na simulação (ponto scope1)

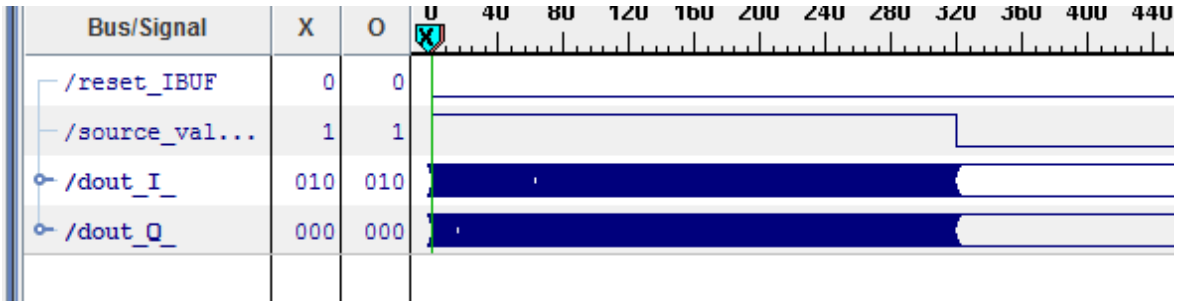

fig. 99 – Saída da memória no *ChipScope* (ponto scope1)

Podemos ver à saída da memória uma trama completa com os valores I, e Q de uma transmissão. Fazendo o zoom das imagens podemos ver o correcto comportamento no hardware.

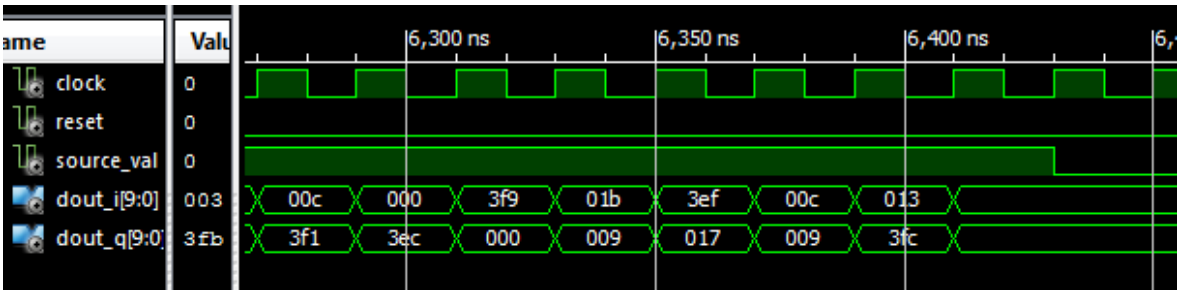

fig. 100 – Vista com zoom da saída da memória, na simulação (ponto scope1)

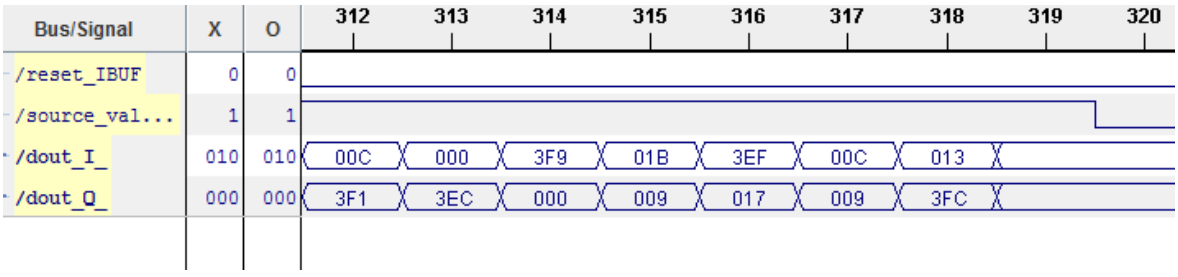

fig. 101 - Vista com zoom da saída da memória, no *ChipScope* (ponto scope1)

Visto que a trama está a entrar correctamente na cadeia, é possível avançar para o ponto seguinte.

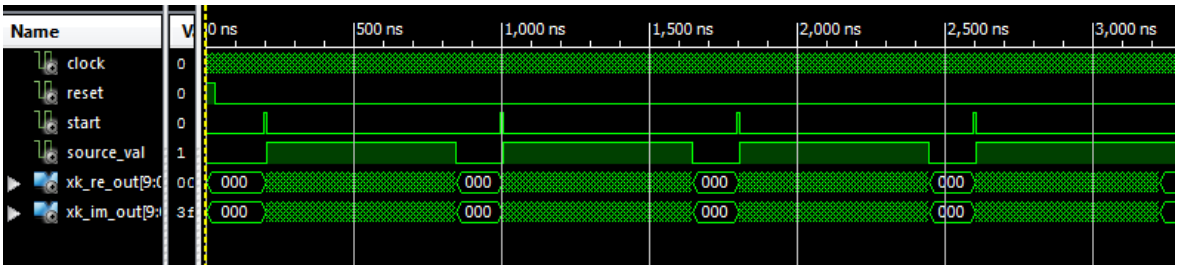

fig. 102 - Saída da bloco de remoção do prefixo cíclico na simulação (ponto scope2)

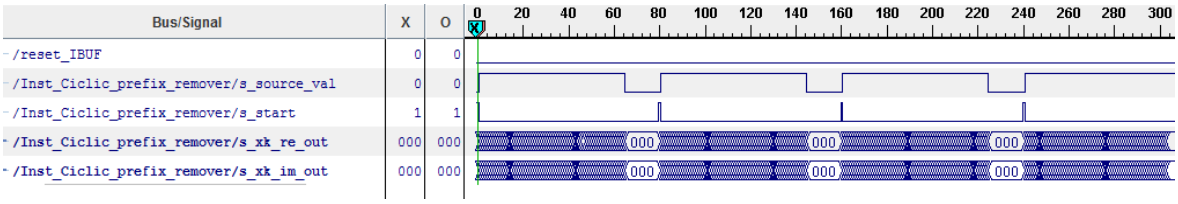

fig. 103 - Saída da bloco de remoção do prefixo cíclico no *ChipScope* (ponto scope2)

Podemos ver o comportamento esperado no hardware, ou seja a remoção da parte correspondente ao prefixo cíclico. Penso que nesta fase não é necessário o zoom, portanto avanço de seguida para a saída da FFT.

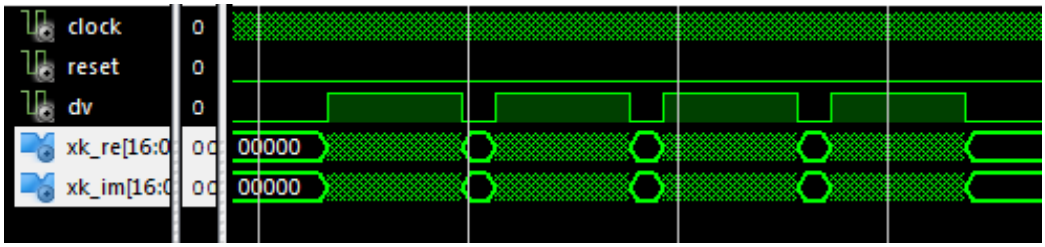

fig. 104 - Saída da bloco de FFT na simulação (ponto scope3)

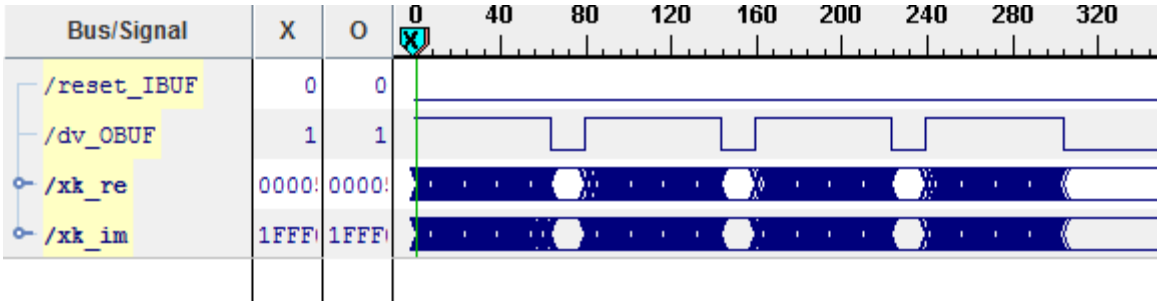

fig. 105 - Saída da bloco de FFT no *ChipScope* (ponto scope3).

Numa vista geral de toda a trama podemos ver os quatro símbolos que constituem a trama que está a ser analisada, calculados na saída da FFT. Para confirmar que estão correctos os cálculos apresenta-se o zoom destas imagens.

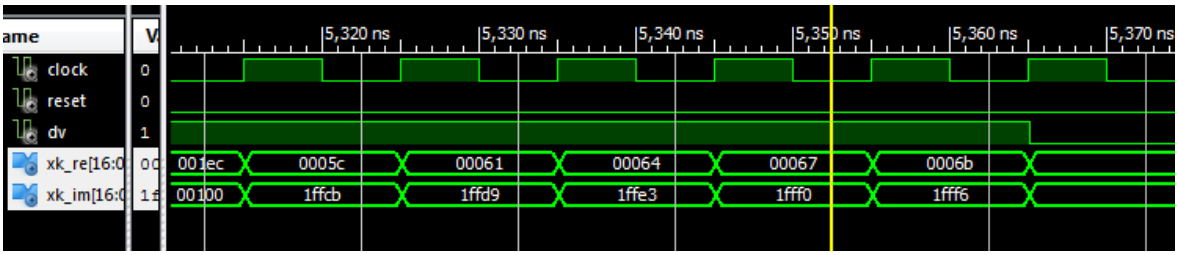

fig. 106 – Vista com zoom da saída da FFT, na simulação (ponto scope3)

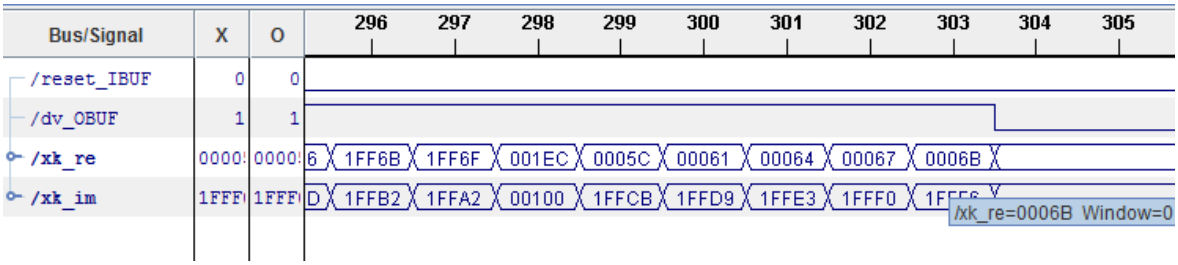

fig. 107 - Vista com zoom da saída da FFT, no *ChipScope* (ponto scope3).

Podemos ver que até este ponto tudo no hardware esta de acordo com o que seria de esperar. Avanço para o próximo último ponto no final da cadeia de recepção.

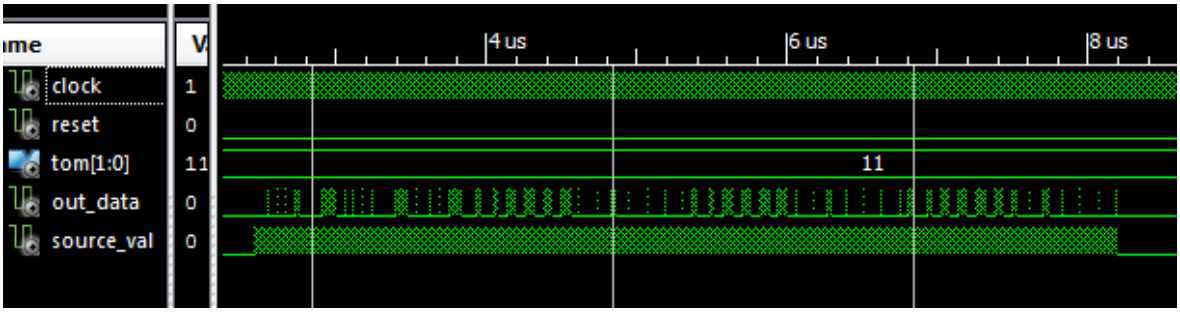

fig. 108 – Vista dos sinais de saída da cadeia de transmissão, em simulação (ponto scope4)

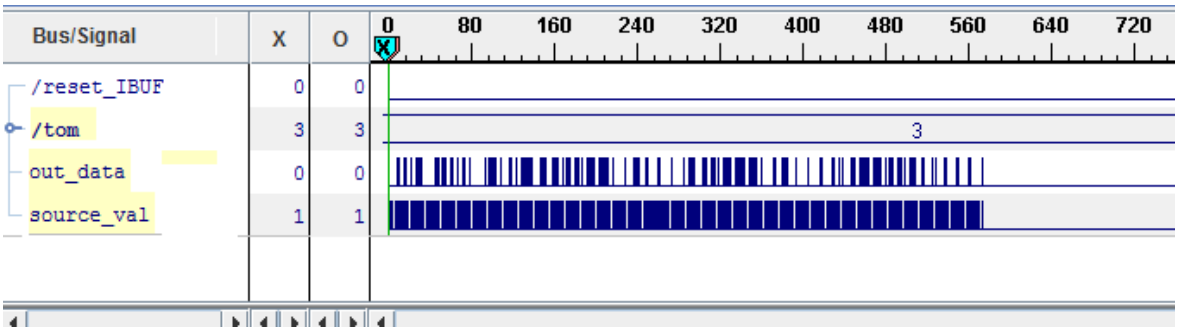

fig. 109 - Vista dos sinais de saída da cadeia de transmissão, no *Chipscope* (ponto scope4)

As imagens em cima mostram a vista completa da trama descodificada na *stream* de bits. Para ver se os bits estão correctos no hardware apresentam-se as imagens com zoom.

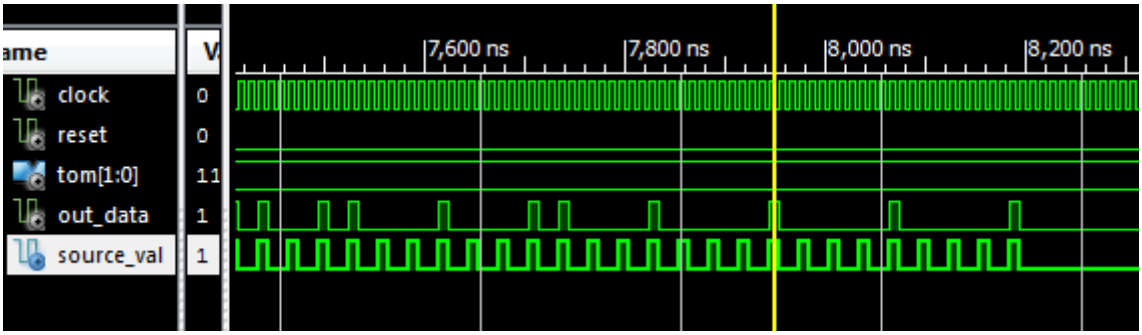

fig. 110 – Zoom da bit stream descodificada na simulação (ponto scope 4)

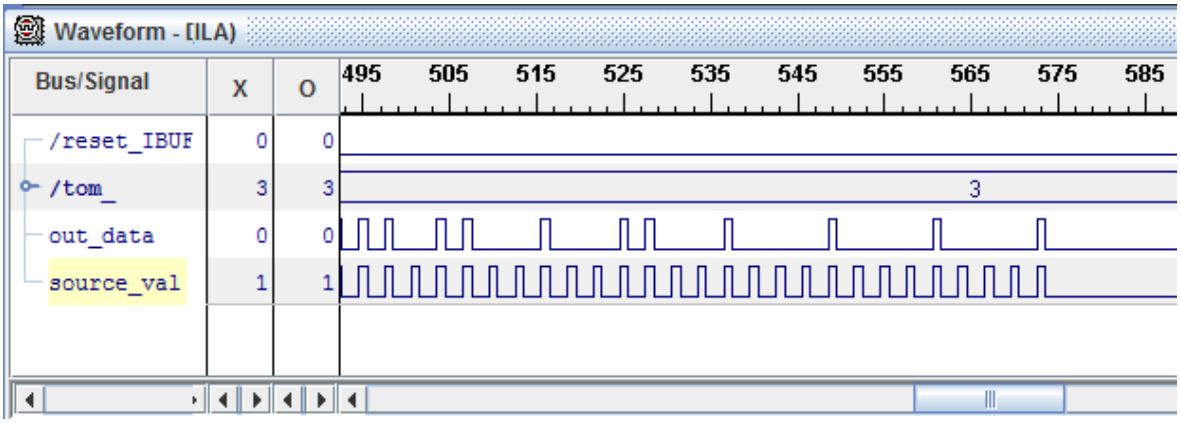

fig. 111 - Zoom da bit *stream* descodificada no *Chipscope* (ponto scope 4)

7 Conclusões e Trabalho Futuro

7.1 Conclusões

.

A minha tarefa neste trabalho foi a de realizar em VHDL para FPGAs da XILINX os blocos de que realizam o mapeamento das portadoras OFDM, a inserção das portadoras piloto e a realização do bloco de IFFT, isto na transmissão. Na recepção os blocos que fazem as operações inversas, o bloco de FFT e de *DeMapping*. Todos os blocos a serem feitos de acordo as especificações da norma. Todos os blocos implementados para a realização destas tarefas foram gradualmente testados em simulação, onde se verificou que o seu comportamento estava de acordo com o pedido pela norma. Depois de testados em simulação, a sua verificação foi feita novamente, desta feita em hardware, onde se mostrou que para a SPARTAN 3E, a trabalhar com um relógio de 50MHz os blocos se comportam como o que se verificou na simulação.

Com este trabalho que é parte integrante num projecto maior, o projecto Headway, espero ter dado um pequeno contributo para o futuro das comunicações veiculares, e do projecto Headway.

7.2 Trabalho futuro

Um dos grandes problemas detectados nesta implementação, é a grande quantidade de memória utilizada, nomeadamente no bloco pilot_insertion. Esta necessidade de um grande fifo na entrada deve-se à diferença no número de amostras que são adicionadas neste bloco. Para resolver esta situação, uma abordagem com dois relógios com ritmos distintos deva ser considerada. Outra solução pode passar pela alteração do processo de *hanshake* entre os blocos da cadeia. As duas soluções em conjunto também podem e devem ser consideradas. E assim fazer a optimização do sistema e dos recursos.

Algo que não está actualmente considerado, é o bloco que faz uma primeira avaliação do campo SIGNAL, na recepção. Isto é algo essencial para a correcta recepção pois a informação de ritmo de transmissão é necessária para muitos dos blocos da recepção.

Ficaram a faltar neste trabalho resultados de valores como os de EVM (*Error Vector Magnitude*) da constelação, de modo a ver se estes cumprem os requisitos da norma. Caso não os cumpram algumas soluções podem ser aplicadas como por exemplo o aumento do número de bits de

entrada da IFFT, de maneira a ter resultados com mais precisão, e uma consequentemente ter mais precisão nos valores de normalização das constelações.

Referências

[1] - http://www.brisainovacao.pt/pt/id/projectos/headway
http://www.brisainnovation.com/en/rd/projects/headway - última visita Outubro 2011

[2] – Hannes Hartenstein, Kenneth P Laberteaux, "VANET: Vehicular Applications and Inter-Networking Technologies", JohnWiley & Sons Ltd, 2010

[3] – Martinez F.J, Cano J.C., Calafate C.T., Manzoni P., "A performance evaluation of warning message dissemination in 802.11p based VANETs", 2009.

[4] –" Advances in Inter-Vehicle Communications Systems and Potential Military Applications"
http://www.science.mod.uk/codex/issue5/journals/documents/codex5_journals_4.pdf - última visita Outubro 2011

[5]- http://www.hindawi.com/journals/ijvt/2010/797405/ - última visita Outubro 2011

[6]-
http://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_estudos&ESTUDOSest_boui=56509088&ESTUDOSmodo=2 – última visita Outubro 2011

[7] - " WLAN 802.11p Measurements for Vehicle to Vehicle (V2V) DSRC Application Note"
http://www2.rohde-schwarz.com/file_12631/1MA152_2e.pdf, última visita Outubro 2011

[8] - http://www.itsmassachusetts.org/pdfs/04mtg/GHz.pdf - última visita Setembro 2011

[9] - http://www.kapsch.net/en/ktc/press/articles/Pages/ktc_100915_pr.aspx - última visita Outubro 2011

[10] - http://www.denso-europe.com/Vehicle-to-Vehicle-Communications---1014820000000001.aspx - última visita Outubro 2011

[11] - J. Mitola. "The Software Radio Architecture". IEEE Communications Magazine, 1995

[12] – Jeffrey H. Reed "Software RADIO: A modern Aproach to Radio Engineering", prentice Hall

[13]- SDR fórum - http://www.wirelessinnovation.org/ - última visita Outubro 2011

[14]-J. Mitola, III, "Cognitive Radio for Flexible Multimedia Communications", Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 IEEE International Workshop on, pp. 3 –10, 1999

[15] - S. Haykin, “Cognitive Radio: Brain-Empowered Wireless Communications,” IEEE Journal, vol. 23, No 2

[16] - http://wwwen.zte.com.cn/endata/magazine/ztecommunications/2009year/no2/articles/200907/t20090701_173473.html, última visita Outubro 2011

[17] - Charan Langton, “Intuitive Guide to Principles of Communications - OFDM”, 2004, www.complextoreal.com/chapters/ofdm2.pdf - última vista Outubro 2011

[18] - Richard van Nee, Ramjee Prasad, “OFDM for Wireless Multimedia Communications”, Artech House, 2000

[19] - L. Hanzo, M. Munster, B.J. Choi, T. Keller - “ OFDM and MC-CDMA for broadband multi-user communications, WLANs, and broadcasting”, John Wiley and Sons, 2003

[20] - Keithley Instruments, “An Introduction to Orthogonal Frequency Division Multiplex Technology”, 2007-2008

[21] - Lawrey Eric Phillip;”Adaptive Techniques for Multiuser OFDM”; Electrical and Computer Engineering School of Engineering James Cook University, 2001

[22] – IEEE Std 802.11™-2007 (Revision of IEEE Std 802.11-1999) “IEEE Standard for Information technology - Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements”

[23] - IEEE Std 802.11p™-2010

[24] – Pong P. Chu, “FPGA Prototyping by VHDL examples – XILINX Spartan-3 version”, Wiley Interscience

[25] – XILINX “LogiCORE IP Fast Fourier Transform v7.1”, abril 19, 2010